PROJECT 2
UBER COMPETITOR DATABASE

Introduction to Database Management Systems (CSE 581)

ARJUN KALYANASUNDARAM

518235238

1. <u>Abstract:</u>

In this Project an **Uber Competitor Database** is created using MS SQL Server Management studio, the database diagram has been constructed using vertableo and the E/R diagram is constructed using creately(online tool).

The database is created with a number of tables and it has been tested with various test cases. The purpose of the database is to bring about a model for an Uber Competitor by bringing in the necessary information for a company to compete with Uber. The database contains critical information of a company's data and the data can be used in a number of ways by using SQL server management studio. The relational database that has been created provides more than basic information for the successful data management for a company in the transportation industry. However, the database is not just limited to the table and more additional features can be added due to the ease of the foreign key constraint in this database. The tables were created, and sample data was entered to test and run the database. In addition to that a number of Transact SQL operations such as stored procedures, functions, triggers, column constraints and a number of views are created and tested on this database. Also, certain Security features have been provided to the database

2.DESIGN

2.1 Introduction

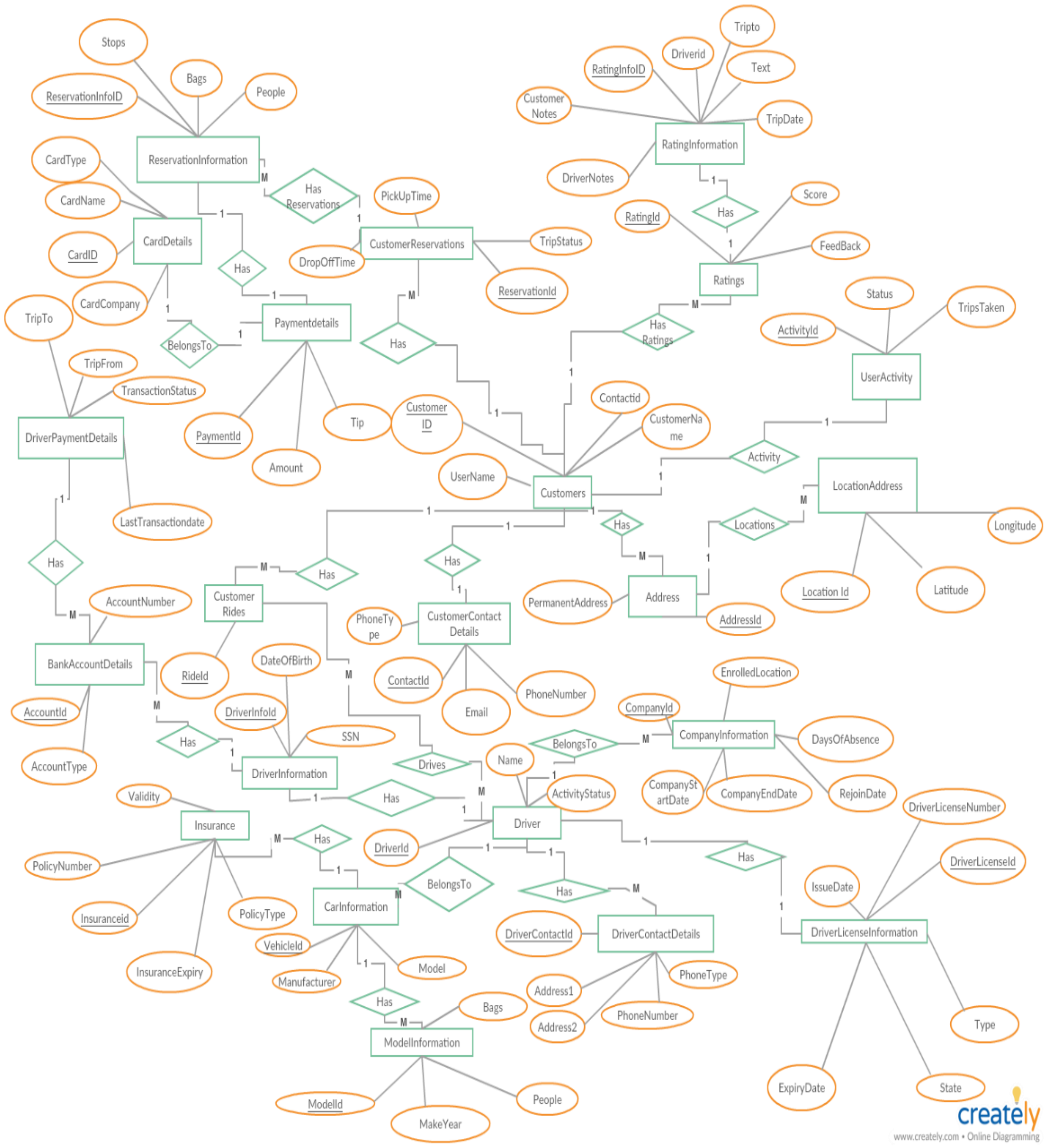
In this project a database for an uber competitor is designed, tested and verified. The purpose of such a database is to obtain the necessary information a company in the transportation field would require. Firstly, the database is designed with a number of essential features and then built upon by adding relevant tables, tables that add critical information to the database. It is ensured that a database with a number of entities should be normalized. Here all these entities provide the vital information to construct a cab database. The competitor will be able to test the database with a number of queries and T-SQL commands. The purpose is to ensure that each table is normalized and is not dependent directly on other entities which is a very crucial factor. The database for the competitor has a number of relationships including one to many, many to many, many to one and one to one. All these relationships provide necessary information and tell how the data is to be stored. Use of such a database increases the efficiency of the business operation and reduces overall costs. The database uses the perfect schema to allow updating and viewing data at ease. Also, one of the reasons why this design supersedes other databases is its security information. Moreover, the database design ensures that redundant information is not stored in it. When less redundant information is stored on the database the database efficiency increases by speed and storage. More the data stored in the memory more difficult it would become to access data. A unique feature of this database is that it gives vital information for each and every entity and its attributes. The design is made in such a way that any important attribute can be added to the particular entity easily using SQL server. Since this database has been designed in SQL server data retrieval and data access have never become easier since. Queries used here are powerful tools that provide information instantly. In this database most of the entities one can look for in the substantial growth of the company has been identified and entered and also newer entities can be added when necessary due the primary key and foreign key constraints this database has provided. The Business problem in this industry is uniquely identified in each domain and then the tables have been constructed to solve these problems. A reason would be to make the database more understandable for the user. Overall the database has been carefully designed, implemented and tested to tackle most of the business problems an uber competitor would endure.

2.2 Design Considerations

The database has been designed by taking into account a number of considerations to solve the given business problem. A total of about 22 tables have been designed and sample data have been input into every table to test it's working, the testing is done using a number of SQL queries and T-SQL commands such as stored procedures, functions, triggers and also with views. The database is normalized to it's 3rd normal form and data can be observed from each table. The following relationships were used in the design one to one, one to many, many to one and many to many. These relationships are clearly shown in the E/R diagram which has been created. Also, constraints for every table have been provided to determine which have null and non-null constraints. Moreover, each table is identified with it's one unique Primary key mainly for the purpose of normalizing it and a number of foreign key constraints have been provided to ensure that the table are related, and no redundancy occurs in the database. The design includes a number of customers and also number of drivers, here customers and drivers have a any to many relationships. The customer has 2 kind of address details one as the permanent address and also a location address which works on coordinated to give the customers location since each customer need not be in a particular address. The customer have the special attribute of checking their payments, give ratings to drivers and also provide some notes on the drivers and for the company's benefit a user activity table is given to find the most frequented customers and also the customers have a table to look up reservations and the reservation ID is uniquely identified and is used to link to other tables such as payment details to look up payment information and card details. Moreover, the customers have their own ratings which are given to them by the drivers. This also allows the drivers to keep tab on any damages that occurred to the car and give them a rating based on customers behavior. Also the reservation information contains details about the number of people who used the cab and the number of bags they had to keep a clear data on customers by trip location. In the drivers table it is uniquely linked to a number of tables to contain necessary information about the driver. A separate table for driver information has been created to store secure information of the driver such as SSN and the bank account the driver is linked to. Each transaction into the bank account comes with the last ride taken and the bank transactions for each ride this allows the driver to keep tab on each transaction by location, which is a unique feature that is needed. In addition to that the driver has a separate table for his contact details and most importantly the vehicle information which is linked to the insurance also. The vehicle information is split into tables to ensure normalization each table has a specific model and the model may vary by year. A driver may have more than one car too. The design considerations have been carefully analyzed and the database has been implemented. A number of successful queries have been tested with the sample data that has been input in the table.

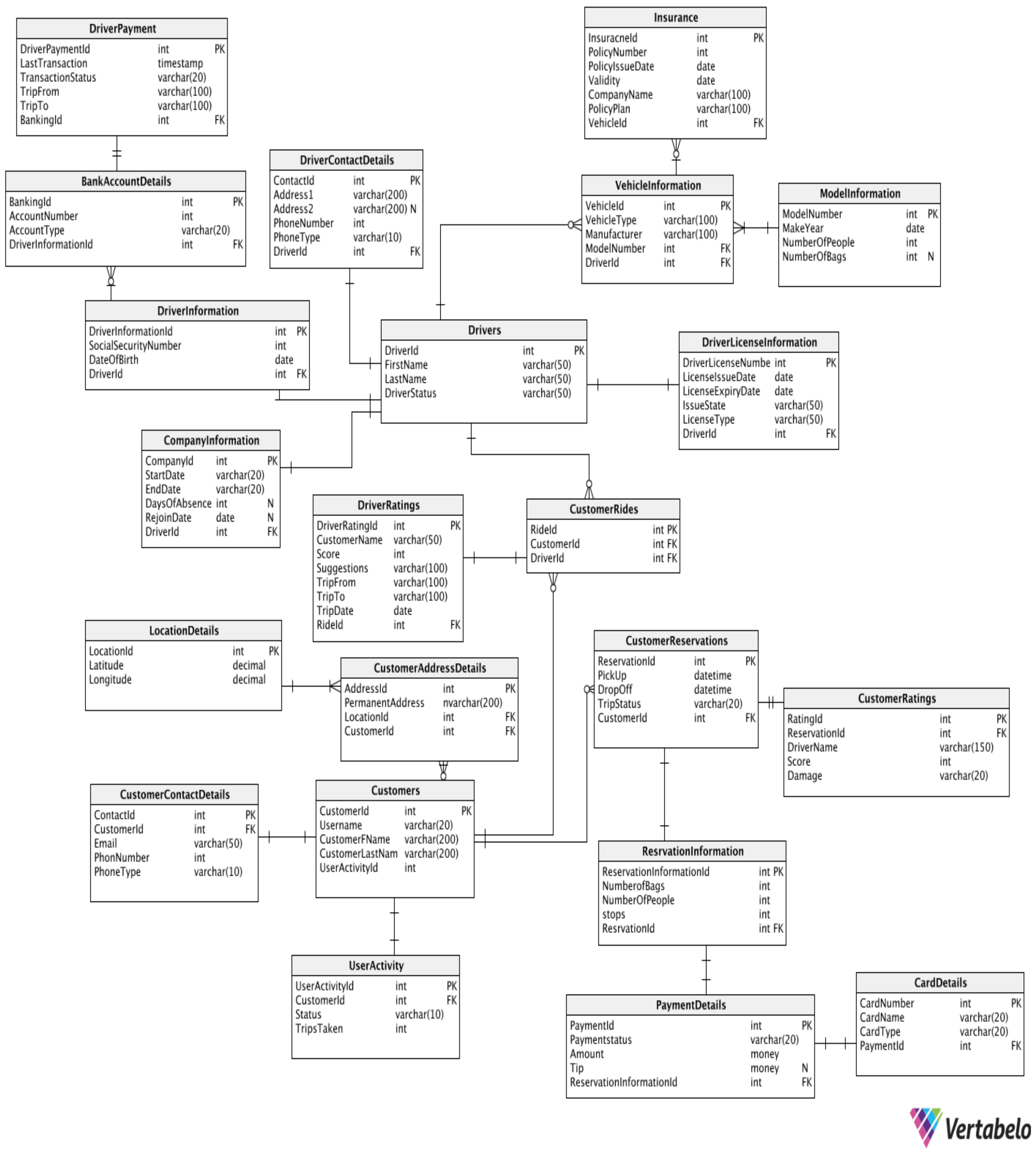
2.3 Entity Relationship Diagram

The entity relationship diagram has been created using the online tool creately



2.4 Database Diagram

The Database Diagram is in it’s third normal form and has been constructed using vertableo.

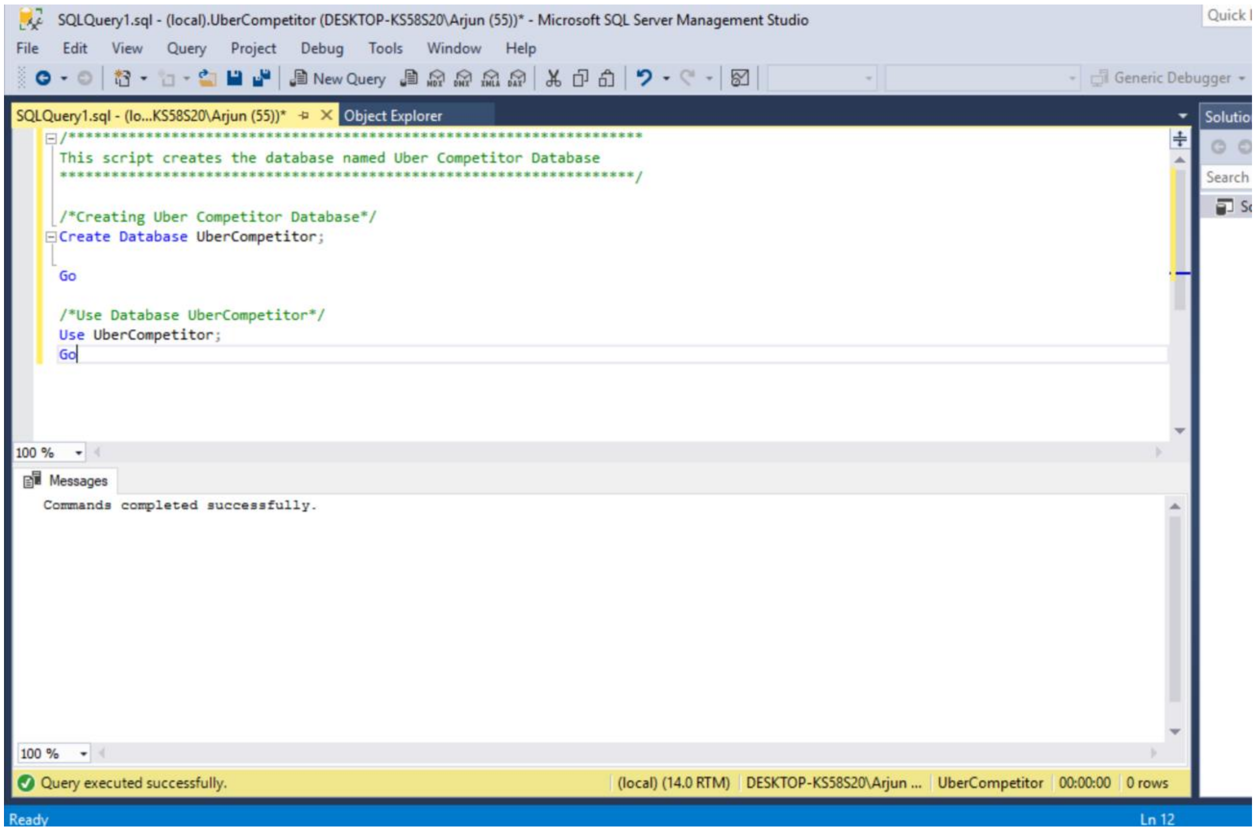


3. Implementation

In the implementation the database has been created using SQL server management studio. The database and tables are created and then sample data is input for the testing of the database.

3.1 Created database named UberCompetitor

```sql
/*******************************************************************
This script creates the database named Uber Competitor Database
*******************************************************************/

/*Creating Uber Competitor Database*/
Create Database UberCompetitor;

Go

/*Use Database UberCompetitor*/
Use UberCompetitor;
Go
```



```
Commands completed successfully.
```

Query executed successfully. | (local) (14.0 RTM) | DESKTOP-KS58S20\Arjun ... | UberCompetitor | 00:00:00 | 0 rows

3.2 Creating the tables:

The necessary tables are created by using the ‘Create Table’ command here the necessary constraints, primary key and foreign key are provided. Also, all the nullability information for each table is included.

```
/* Table Drivers*/
Create Table Drivers
(
DriverId    int  Not NULL Primary Key,
FirstName   varchar(50) Not NULL,
LastName    varchar(50) Not NULL,
DriverStatus   varchar(50) Not NULL,
);

/* Table DriverLicenseInformation */
Create Table DriverLicenseInformation
(
DriverLicenseNumber    int Not NULL Primary Key,
LicenseIssueDate       date Not NULL,
LicenseExpiryDate      date Not NULL,
IssueState            varchar(50) Not NULL,
LicenseType           varchar(50) Not NULL,
DriverId              int NOT NULL Foreign Key References Drivers(DriverId),
);



/* Table DriverInformation */
Create Table DriverInformation
(
DriverInformationId    int  Not NULL Primary Key,
SocialSecurityNumber    int Not Null,
DateOfBirth             date Not Null,
DriverId              int NOT NULL Foreign Key References Drivers(DriverId),
);



/* Table ModelInformation*/
Create Table ModelInformation
(
ModelNumber    int  Not NULL Primary Key,
MakeYear       date   Not Null,
NumberOfPeople    int  Not Null,
NumberOfBags int Null,
);
```

```
/* Table CompanyInformation*/

Create Table CompanyInformation
(
CompanyId  int  Not NULL Primary Key,
StartDate  date   Not Null,
EndDate     date NULL,
DaysOfAbsence int Null,
RejoinDate  date Null,
DriverId   int NOT NULL Foreign Key References Drivers(DriverId),
);

/* Table VehicleInformation*/
Create Table VehicleInformation
(
VehicleId  int  Not NULL Primary Key,
VehicleType  varchar(100)  Not Null,
Manufacturer  varchar(100) Not Null,
ModelNumber int NOT NULL Foreign Key References ModelInformation(ModelNumber),
DriverId int NOT NULL Foreign Key References Drivers(DriverId),
);

/* Table Insurance*/
Create Table Insurance
(
InsuranceId  int  Not NULL Primary Key,
PolicyNumber int  Not Null,
PolicyIssueDate date  Not Null,
Validity date Not Null,
CompanyName varchar(200) Not Null,
PolicyPlan varchar(200) Not Null,
VehicleId int NOT NULL Foreign Key References VehicleInformation(VehicleId),
);


/* Table DriverContactDetails */
Create Table DriverContactDetails
(
ContactId  int  Not NULL Primary Key,
Address1       varchar(200)  Not Null,
Address2     varchar(200)  Not Null,
PhoneNumber   int Not Null,
PhoneType     varchar(10) Not Null,
DriverId     int NOT NULL Foreign Key References Drivers(DriverId),
);

/* Table BankAccountDetails*/
Create Table BankAccountDetails
(
BankingId  int  Not NULL Primary Key,
AccountNumber int  Not Null,
AccountType     varchar(20)  Not Null,
DriverInformationId   int NOT NULL Foreign Key References
DriverInformation(DriverInformationId ),
);
```

```sql
/* Table Customers*/
CREATE TABLE Customers (
    CustomerId int PRIMARY KEY  NOT NULL,
    Username varchar(20)  NOT NULL
    CustomerFName varchar(200)  NOT NULL,
    CusromerLastName varchar(200)  NOT NULL,
    UserActivityId int  NOT NULL,

);

/* Table CustomerRides*/
CREATE TABLE CustomerRides (
    RideId int PRIMARY KEY NOT NULL,
    CustomerId int  NOT NULL Foreign Key References Customers(CustomerId),
    DriverId int NOT NULL Foreign Key References   Drivers(DriverId),

);

/* Table CustomerReservations*/
CREATE TABLE CustomerReservations (
    ReservationId int PRIMARY KEY   NOT NULL,
    TripStatus varchar(20)  NOT NULL,
    CustomerId int  NOT NULL Foreign Key References Customers(CustomerId),
    PickUp datetime  NOT NULL,
    DropOff datetime  NOT NULL,


);

/* Table CustomerRatings*/
CREATE TABLE CustomerRatings (
    RatingId int PRIMARY KEY   NOT NULL,
   ReservationID int  NOT NULL Foreign Key References
CustomerReservations(ReservationId),
    DriverName varchar(150)  NOT NULL,
    Score int  NOT NULL,
    Damage varchar(20)  NOT NULL,

);


/* Table CustomerContactDetails*/
CREATE TABLE CustomerContactDetails (
    ContactId int  PRIMARY KEY NOT NULL,
    CustomerId int  NOT NULL Foreign Key References Customers(CustomerId),
    Email varchar(50)  NOT NULL,
    PhonNumber int  NOT NULL,
    PhoneType varchar(10)  NOT NULL,

);

/* Table LocationDetails*/
CREATE TABLE LocationDetails (
    LocationId int PRIMARY KEY NOT NULL,
    Latitude decimal  NOT NULL,
    Longitude decimal  NOT NULL,
);
```

```
/* Table CustomerAddressDetails*/
CREATE TABLE CustomerAddressDetails
(
    AddressId int PRIMARY KEY NOT NULL,
    PermanentAddress varchar(200)  NOT NULL,
    LocationId int  NOT NULL Foreign Key References LocationDetails(LocationId),
    CustomerId int  NOT NULL Foreign Key References Customers(CustomerId),

);
```

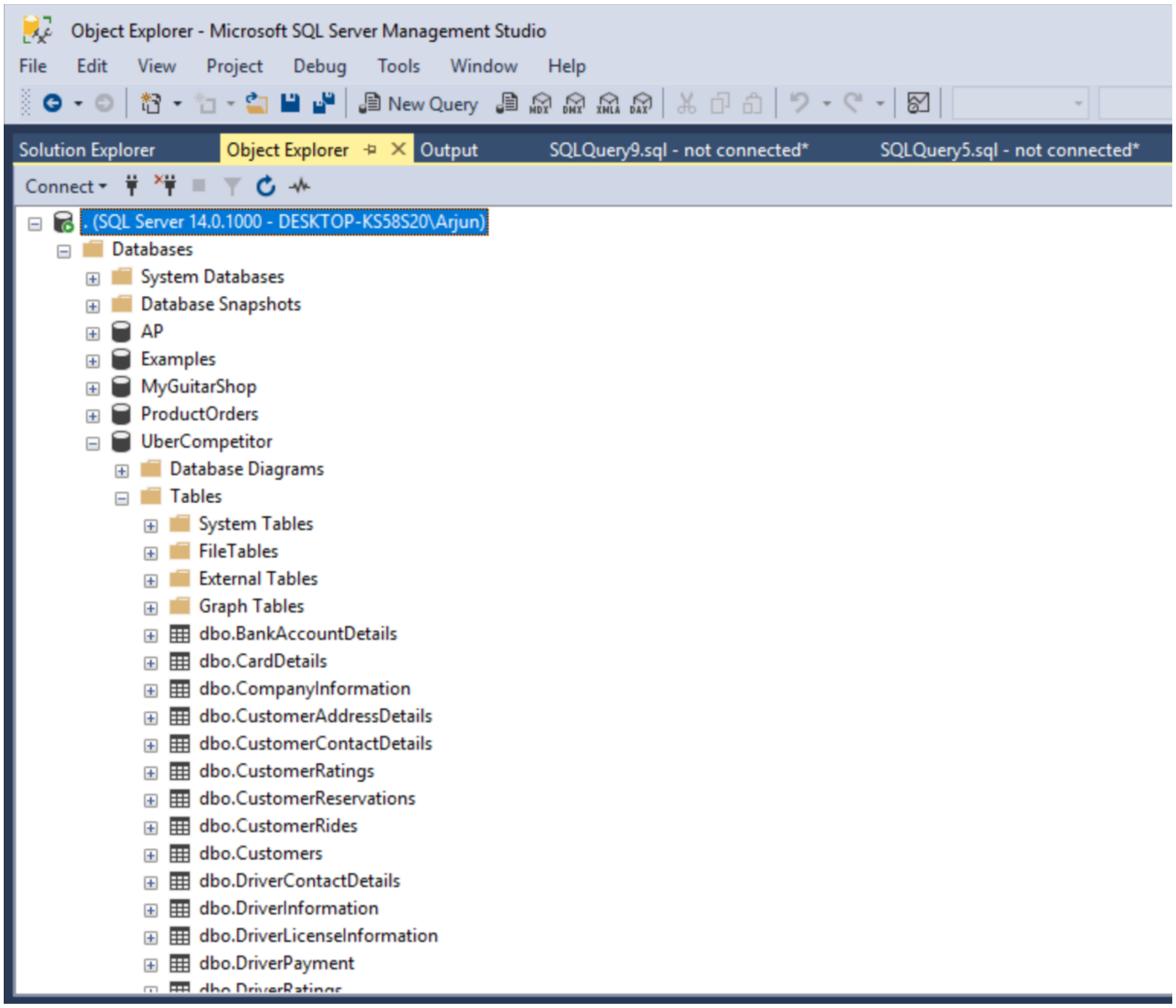
```
/* Table ReservationInformation*/
CREATE TABLE ResrvationInformation (
    ReservationInformationId int PRIMARY KEY  NOT NULL,
    NumberofBags int  NOT NULL,
    NumberOfPeople int  NOT NULL,
    stops int  NOT NULL,
   ReservationId int  NOT NULL Foreign Key References
CustomerReservations(ReservationId),

);
```

```
/* Table  UserActivity*/
CREATE TABLE UserActivity
 (
    UserActivityId int PRIMARY KEY NOT NULL,
    CustomerId int  NOT NULL Foreign Key References Customers(CustomerId),
    Status varchar(10)  NOT NULL,
    TripsTaken int  NOT NULL,

);
```

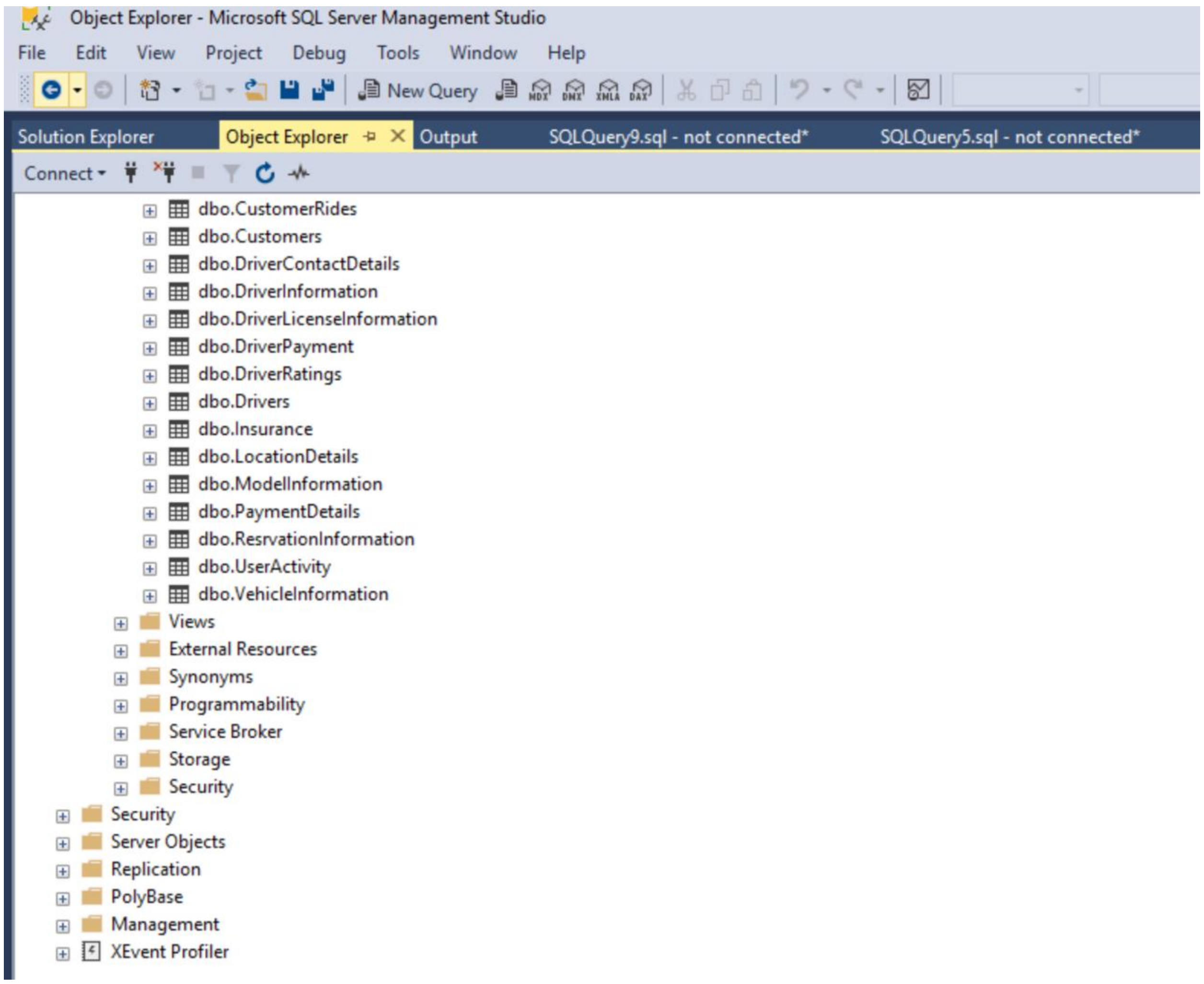
```
/* Table PaymentDetails*/
CREATE TABLE PaymentDetails(
    PaymentId int  PRIMARY KEY NOT NULL,
    Paymentstatus varchar(20)  NOT NULL,
    Amount money  NOT NULL,
    Tip money  NULL,
     ReservationInformationId  int  NOT NULL Foreign Key References
ResrvationInformation(ReservationInformationId),

);
```

```
/* Table CardDetails*/
CREATE TABLE CardDetails
(
    CardNumber int  Primary Key NOT NULL,
    CardName varchar(20)  NOT NULL,
    CardType varchar(20)  NOT NULL,
    PaymentId   int NOT NULL Foreign Key References PaymentDetails( PaymentId ),

);



/* Table DriverPayment*/
CREATE TABLE DriverPayment (
    DriverPaymentId int PRIMARY KEY NOT NULL
    TransactionStatus varchar(20)  NOT NULL,
    TripFrom varchar(100)  NOT NULL,
    TripTo varchar(100)  NOT NULL,
  BankingId int  NOT NULL Foreign Key References BankAccountDetails(BankingId),
    LastTransaction datetime  NOT NULL,

);


/* Table DriverRatings*/
CREATE TABLE DriverRatings (
    DriverRatingId int PRIMARY KEY  NOT NULL,
    CustomerName varchar(50)  NOT NULL,
    Score int  NOT NULL,
    Suggestions varchar(100)  NOT NULL,
    TripFrom varchar(100)  NOT NULL,
    TripTo varchar(100)  NOT NULL,
    TripDate date  NOT NULL,
    RideId int  NOT NULL Foreign Key References CustomerRides(RideId),
);
```

3.3 Screenshots:


Object Explorer - Microsoft SQL Server Management Studio

```
. (SQL Server 14.0.1000 - DESKTOP-KS58S20\Arjun)
  Databases
    System Databases
    Database Snapshots
    AP
    Examples
    MyGuitarShop
    ProductOrders
    UberCompetitor
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
        dbo.BankAccountDetails
        dbo.CardDetails
        dbo.CompanyInformation
        dbo.CustomerAddressDetails
        dbo.CustomerContactDetails
        dbo.CustomerRatings
        dbo.CustomerReservations
        dbo.CustomerRides
        dbo.Customers
        dbo.DriverContactDetails
        dbo.DriverInformation
        dbo.DriverLicenseInformation
        dbo.DriverPayment
        dbo.DriverRatings
```


Object Explorer - Microsoft SQL Server Management Studio
File
Edit
View
Project
Debug
Tools
Window
Help
New Query
Solution Explorer
Object Explorer
Output
SQLQuery9.sql - not connected*
SQLQuery5.sql - not connected*
Connect

dbo.CustomerRides
dbo.Customers
dbo.DriverContactDetails
dbo.DriverInformation
dbo.DriverLicenseInformation
dbo.DriverPayment
dbo.DriverRatings
dbo.Drivers
dbo.Insurance
dbo.LocationDetails
dbo.ModelInformation
dbo.PaymentDetails
dbo.ResrvationInformation
dbo.UserActivity
dbo.VehicleInformation
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
Security
Server Objects
Replication
PolyBase
Management
XEvent Profiler

4. Testing of the Database

4.1 Inserting values

Each table contains 5 rows of sample values for the testing of the database

```
USE UberCompetitor

Insert Into Drivers values
('1','Carlos','John','Active'),
('2','Robert','jan','Active'),
('3','David','Katen','Active'),
('4','Amanda','Johnson','InActive'),
('5','Bridget','Paul','InActive');



Insert Into DriverLicenseInformation values
('1000001','2010-11-22','2022-11-22','CA','CAR','1'),
('1000002','2011-10-21','2021-10-21','NJ','CAR','2'),
('1000003','2014-08-11','2024-08-11','NY','ALL','3'),
('1000004','2013-07-11','2023-08-11','CT','CAR','4'),
('1000005','2015-05-10','2025-05-22','WA','CAR','5');

Insert Into DriverInformation values
('1001','1117483247','1982-09-22','1'),
('1002','1147283642','1980-01-20','2'),
('1003','2147333343','1985-03-15','3'),
('1004','1142482643','1972-02-13','4'),
('1005','1147483642','1990-11-22','5');

Insert Into ModelInformation values
('3121','2010-01-22','6','8'),
('3123','2012-02-11','4','6'),
('3124','2002-10-11','4','6'),
('3125','2019-02-10','6','8'),
('3126','2017-01-12','6','8');

Insert Into CompanyInformation values
('451','2010-11-22','2011-11-13','3','2012-11-22','1'),
('452','2009-11-22','2010-10-12','10','2013-09-14','2'),
('453','2008-02-13','2011-12-12','15','2014-01-11','3'),
('454','2003-11-11','2010-10-12','22','2015-09-14','4'),
('455','2005-11-10','2014-10-12','45','2018-09-14','5')



Insert Into VehicleInformation values
('2001','Car','FORD','3121','1'),
('4442','MiniVan','DODGE','3123','2'),
('3331','Car','CHEVROLET','3124','3'),
('3113','Car','BMW','3125','4'),
('2213','Car','AUDI','3126','5');
```

```
Insert Into Insurance  values
('93331','112321','2018-11-10','2019-11-10','Prudential','FullCover','2001'),
('93332','112324','2018-09-05','2019-09-05','Aetna','FullCover','4442'),
('93213','312332','2018-08-05','2019-08-05','Life','PartialCover','3331'),
('29991','133233','2018-02-05','2019-02-05','Life','PartialCover','3113'),
('19388','111113','2018-03-07','2019-03-07','Prudential','FullCover','2213');
```

```
Insert Into DriverContactDetails values
('1221','322 Ackerman Avenue Binghamton NY33321','443 Lancaster Avenue Binghamton
NY32221','21112222','Mobile','1'),
('1222','222 Lexington Avenue Binghamton NY33392','222 George Avenue Binghamton
NY32211','11112222','Work','2'),
('1223','921 Euclid  Avenue SanJose CA118810','Light Street Avenue SanJose
CA23145','13119222','Work','3'),
('1224','182 East VanNess SanFrancsico CA11822','932 South VanNess SanFrancsico
CA11224','14118222','Work','4'),
('1225','336 Lawn Avenue Summit NJ08972','223 Fall Street
Milburn','13114212','Mobile','5');
```

```
Insert Into BankAccountDetails Values
('2221','1111133','Savings','1001'),
('2233','2333922','Checking','1002'),
('2122','1334442','Savings','1003'),
('3222','2333113','Savings','1004'),
('2254','1333123','Checinng','1005');
```

```
Insert Into Customers Values
('5121','arjkkk','Arjun','Kalyan','9022'),
('5122','randall','Rand','Joel','9023'),
('5123','johndoee','John','Doe','9024'),
('5124','lindaaa','Linda','Price','9025'),
('5125','shawno','Shawn','Watson','9026')
```

```
Insert Into CustomerRides Values
('5671','5121','1'),
('5672','5122','2'),
('5673','5123','3'),
('5674','5124','4'),
('5675','5125','5')
```

```
Insert Into CustomerReservations Values
('391','Confirmed','5121','2019-02-12 17:32:12','2019-02-12 18:32:12'),
('392','Confirmed','5122','2019-01-10 00:32:12','2019-01-10 01:22:05'),
('393','Pending','5123','2019-05-04 01:32:12','2019-05-04 02:20:24'),
('394','Confirmed','5124','2019-03-23 18:22:12','2019-03-23 19:32:12'),
('395','Confirmed','5125','2019-04-23 13:12:10','2019-04-23 13:45:23')
```

```
Insert Into CustomerRatings Values
('291','391','Carlos','5','No Damage'),
('292','392','Robert','4','No Damage'),
('293','393','David','2','No Damage'),
('294','394','Amanda','3','No Damage'),
('295','395','Bridget','5','No Damage')
```

```
Insert into CustomerContactDetails Values
('651','5121','kaln@gmail.com','14555553','Work'),
('652','5122','randall@gmail.com','11248272','Home'),
('653','5123','jdoe@gmail.com','13448222','Work'),
('654','5124','linda@gmail.com','11442231','Home'),
('655','5125','shawn@gmail.com','12942641','Work')
```

```
Insert into LocationDetails Values
('221','22465.777','55557.998'),
('222','22255.745','11123.432'),
('223','45665.982','43792.098'),
('224','98354.674','48748.432'),
('225','18492.474','18965.432')
```

```
Insert into CustomerAddressDetails  Values
('871','14 Crest Avenue Binghamton NY31222 ','221','5121'),
('872','22  Euclid street Binghamton NY31219 ','222','5122'),
('873','74  clinton street SF Downtown CA32229 ','223','5123'),
('874','98  George street SanJose  CA455532','224','5124'),
('875','35  Von Avenue ShortHills  NJ25667','225','5125')
```

```
Insert into ResrvationInformation Values
('465','3','4','0','391'),
('466','1','1','0','392'),
('467','2','2','1','393'),
('468','4','2','0','394'),
('469','3','1','2','395')
```

```
Insert Into UserActivity Values
('654','5121','Active','10'),
('655','5122','InActive','2'),
('656','5123','Active','45'),
('657','5124','Active','23'),
('658','5125','Active','30')
```

```
Insert Into PaymentDetails Values
('2376','Pending','$25','$3','465'),
('2378','Paid','$20','$1','466'),
('2379','Paid','$30','$3','467'),
('2400','Paid','$40','$5','468'),
('2401','Pending','$32','$5','469')



Insert Into CardDetails Values
('111133332','Arjun','Visa','2376'),
('215163752','Randall','Amex','2378'),
('214123142','John','MasterCard','2379'),
('212183312','Linda','Amex','2400'),
('216127121','Shawn','Visa','2401')



Insert Into DriverPayment Values
('892','P','George','Lowfield','2221','2019-02-12 17:32:12'),
('893','N','StHills','Summit','2233','2019-01-10 00:35:12'),
('894','N','SouthVane','EastVa','2122','2019-05-03 13:20:12'),
('895','P','SanJose','Sunnyvale','3222','2019-03-23 18:25:12'),
('896','P','EastOrange','Newark','2254','2019-04-23 13:15:10')



Insert Into DriverRatings Values
('901','Arjun','5','Good','George','Lowfield','2019-02-12','5671'),
('902','Randall','3','Car not clean','StHills','Summit','2019-01-10','5672'),
('903','John','4','Driver too fast','SouthVane','EastVa','2019-05-03','5673'),
('904','Linda','5','Good','SanJose','SunnyVale','2019-03-23 ','5674'),
('905','Shawn','5','Good','EastOrange','Newark','2019-04-23 ','5675')
```

4.2 Stored Procedures

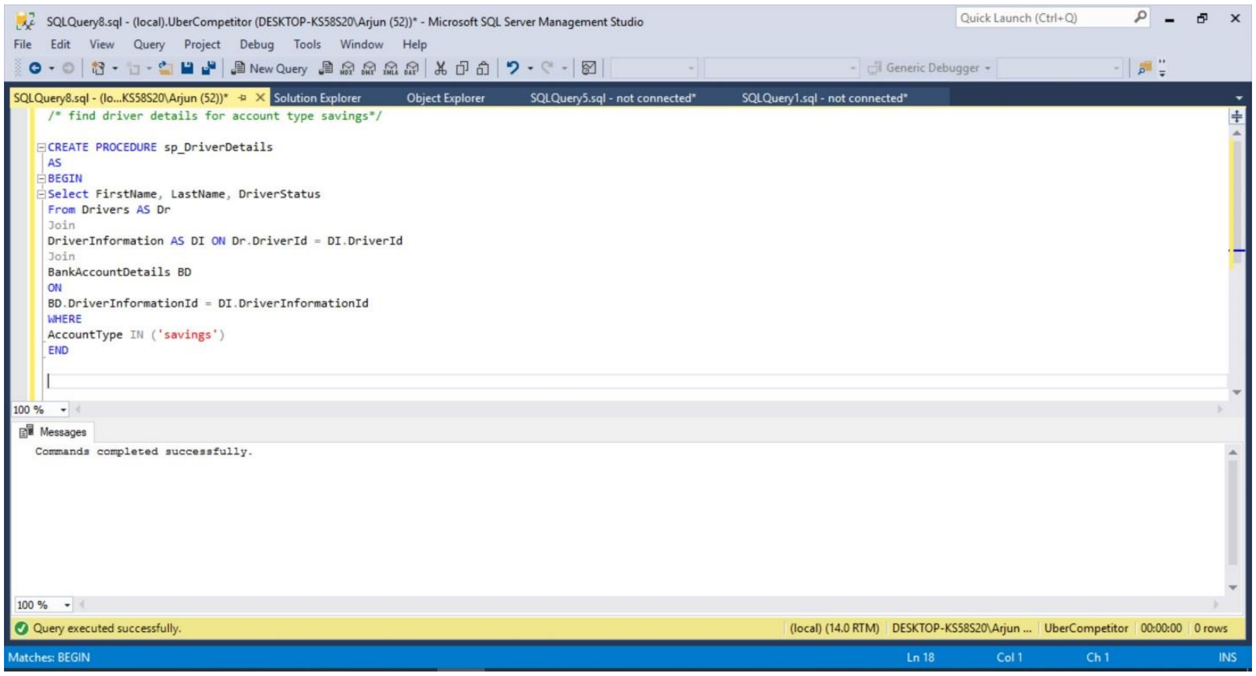
Here 2 stored procedures are created and then executed.

The stored Procedure below finds the details of the driver who have a savings account.

Source Code:

```sql
/* find driver details for account type savings*/

CREATE PROCEDURE sp_DriverDetails
AS
BEGIN
Select FirstName, LastName, DriverStatus
From Drivers AS Dr
Join
DriverInformation AS DI ON Dr.DriverId = DI.DriverId
Join
BankAccountDetails BD
ON
BD.DriverInformationId = DI.DriverInformationId
WHERE
AccountType IN ('savings')
END
```

Screenshot:


SQLQuery8.sql - (local).UberCompetitor (DESKTOP-KS58S20\Arjun (52))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File
Edit
View
Query
Project
Debug
Tools
Window
Help
New Query
Generic Debugger
SQLQuery8.sql - (lo...KS58S20\Arjun (52))*
Solution Explorer
Object Explorer
SQLQuery5.sql - not connected*
SQLQuery1.sql - not connected*
100 %
Messages

```sql
/* find driver details for account type savings*/

CREATE PROCEDURE sp_DriverDetails
AS
BEGIN
Select FirstName, LastName, DriverStatus
From Drivers AS Dr
Join
DriverInformation AS DI ON Dr.DriverId = DI.DriverId
Join
BankAccountDetails BD
ON
BD.DriverInformationId = DI.DriverInformationId
WHERE
AccountType IN ('savings')
END
```

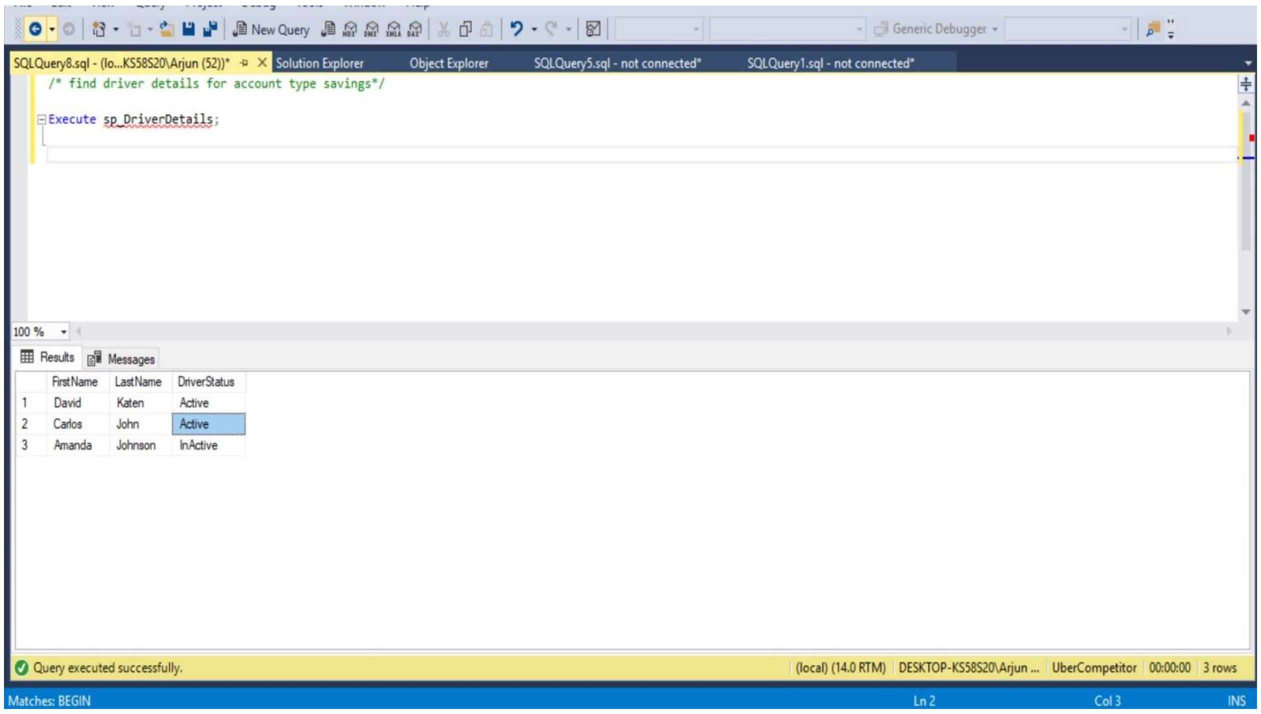
```
Commands completed successfully.
```

```
Query executed successfully.    (local) (14.0 RTM) | DESKTOP-KS58S20\Arjun ... | UberCompetitor | 00:00:00 | 0 rows
Matches: BEGIN    Ln 18    Col 1    Ch 1    INS
```

Source Code:

```
Execute sp_DriverDetails;
```

Screenshot:



```
/* find driver details for account type savings*/

Execute sp_DriverDetails;
```

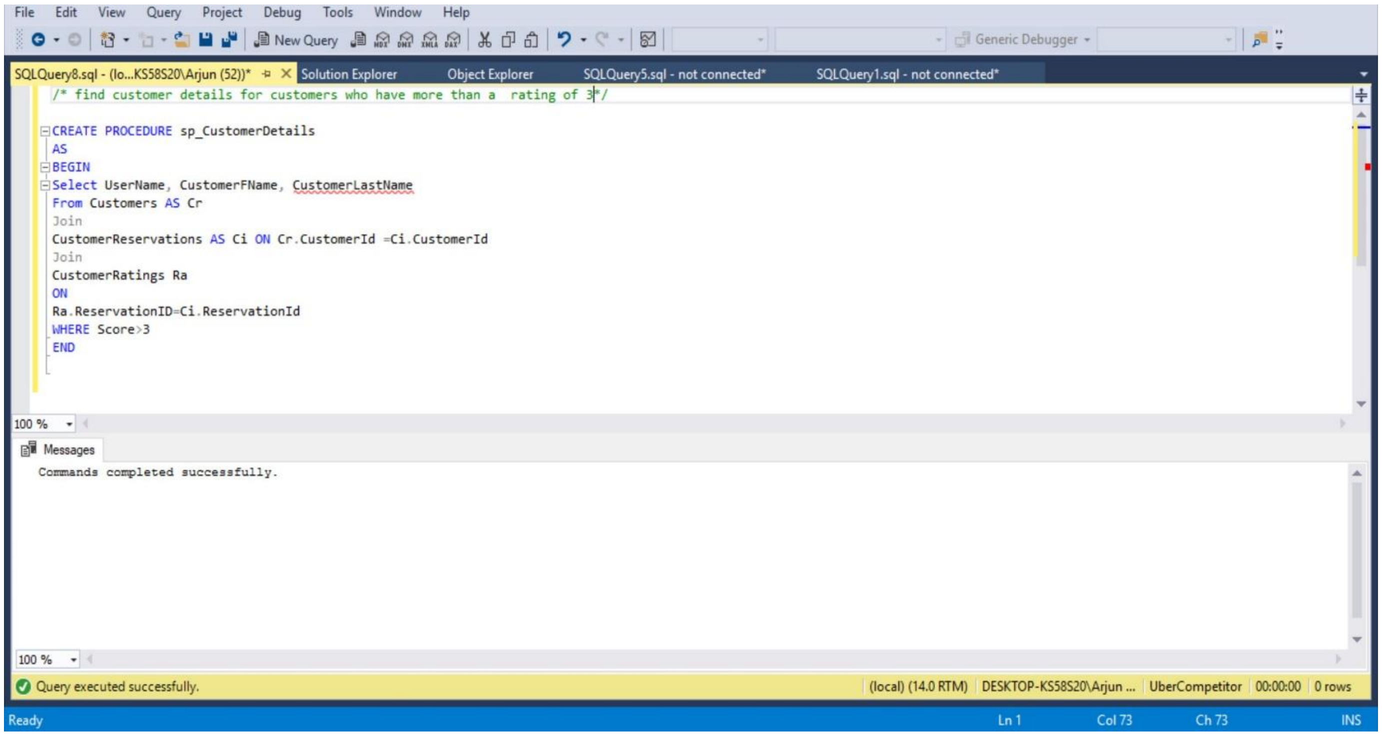
	FirstName	LastName	DriverStatus
1	David	Katen	Active
2	Carlos	John	Active
3	Amanda	Johnson	InActive

Query executed successfully. (local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 3 rows

Source Code:

```
/* finds customer details for customers who have more than a rating of 3*/
CREATE PROCEDURE sp_CustomerDetails
AS
BEGIN
Select UserName, CustomerFName, CustomerLastName
From Customers AS Cr
Join
CustomerReservations AS Ci ON Cr.CustomerId =Ci.CustomerId
Join
CustomerRatings Ra
ON
Ra.ReservationID=Ci.ReservationId
WHERE Score>3
END
```

Screenshot:


File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
SQLQuery8.sql - (lo...KS58S20\Arjun (52))*
Solution Explorer
Object Explorer
SQLQuery5.sql - not connected*
SQLQuery1.sql - not connected*
100 %
Messages
100 %
Ready
Ln 1
Col 73
Ch 73
INS

```
/* find customer details for customers who have more than a  rating of 3*/

CREATE PROCEDURE sp_CustomerDetails
AS
BEGIN
Select UserName, CustomerFName, CustomerLastName
From Customers AS Cr
Join
CustomerReservations AS Ci ON Cr.CustomerId =Ci.CustomerId
Join
CustomerRatings Ra
ON
Ra.ReservationID=Ci.ReservationId
WHERE Score>3
END
```

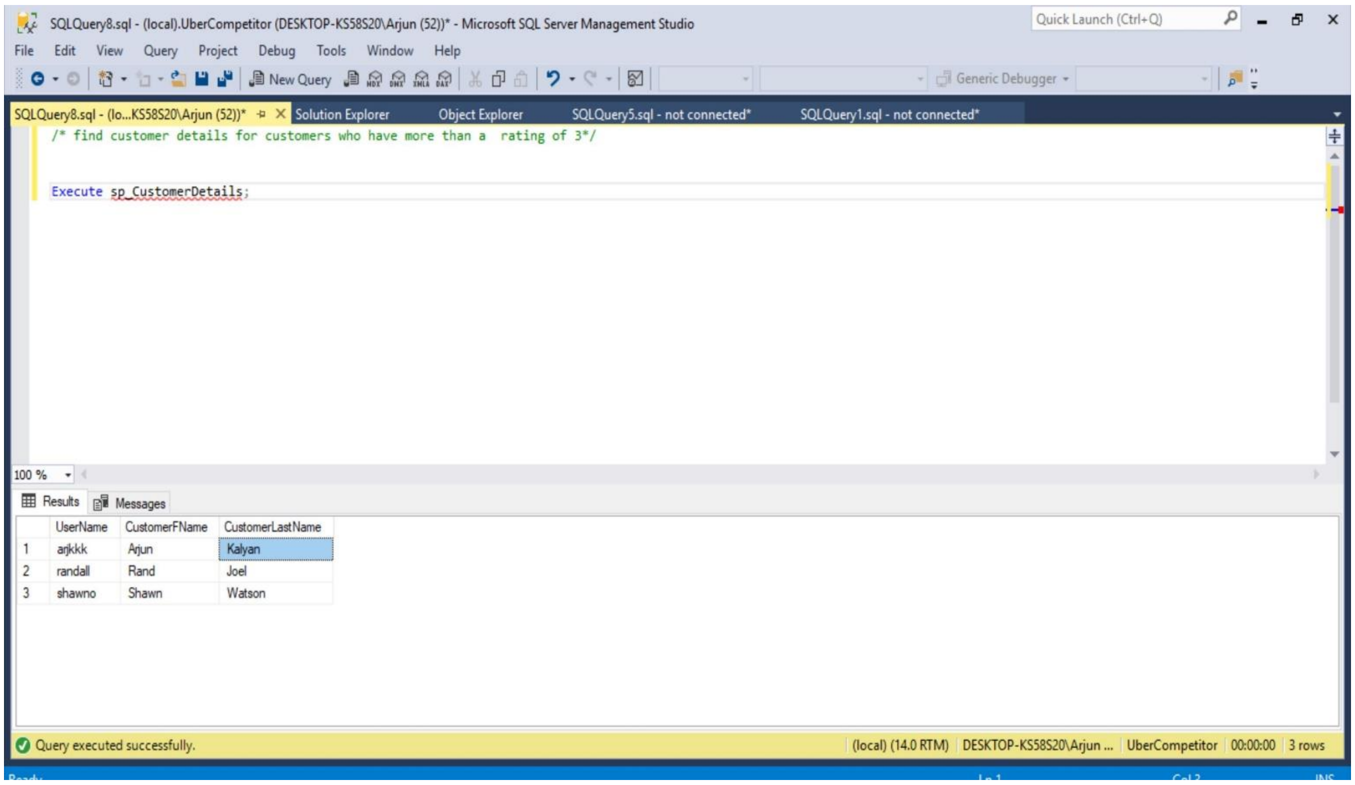
```
Commands completed successfully.
```

Query executed successfully. (local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 0 rows

Source Code:

```
Execute sp_CustomerDetails;
```

Screenshot:



```
/* find customer details for customers who have more than a  rating of 3*/

Execute sp_CustomerDetails;
```

	UserName	CustomerFName	CustomerLastName
1	arjkkk	Arjun	Kalyan
2	randall	Rand	Joel
3	shawno	Shawn	Watson

Query executed successfully. | (local) (14.0 RTM) | DESKTOP-KS58S20\Arjun ... | UberCompetitor | 00:00:00 | 3 rows

4.3 Functions

Functions are created and tested in the database

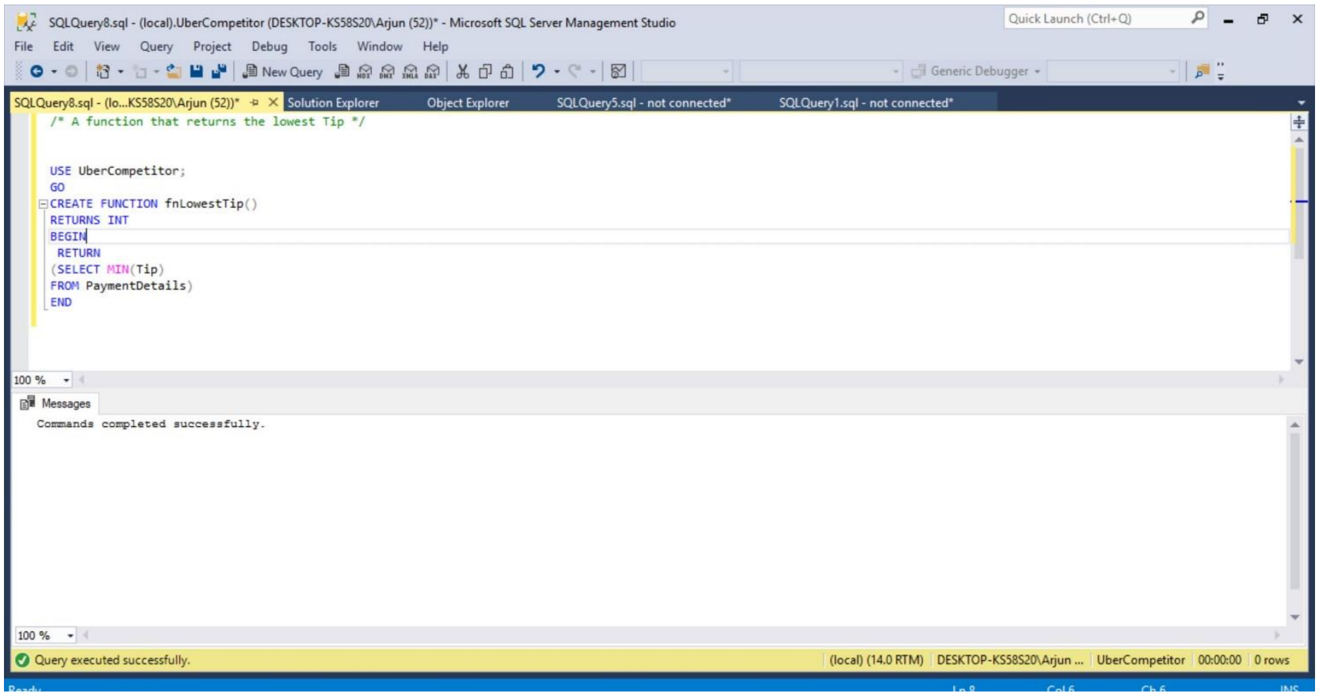
4.31

The function finds the minimum tip given.

Source Code:

```
USE UberCompetitor;
GO
CREATE FUNCTION fnLowestTip()
RETURNS INT
BEGIN
 RETURN
(SELECT MIN(Tip)
FROM PaymentDetails)
END
```

Screenshot



SQLQuery8.sql - (local).UberCompetitor (DESKTOP-KS58S20\Arjun (52))* - Microsoft SQL Server Management Studio
File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
SQLQuery8.sql - (lo...KS58S20\Arjun (52))*
Solution Explorer
Object Explorer
SQLQuery5.sql - not connected*
SQLQuery1.sql - not connected*
100 %
Messages

```
/* A function that returns the lowest Tip */


USE UberCompetitor;
GO
CREATE FUNCTION fnLowestTip()
RETURNS INT
BEGIN
 RETURN
(SELECT MIN(Tip)
FROM PaymentDetails)
END
```

```
Commands completed successfully.
```

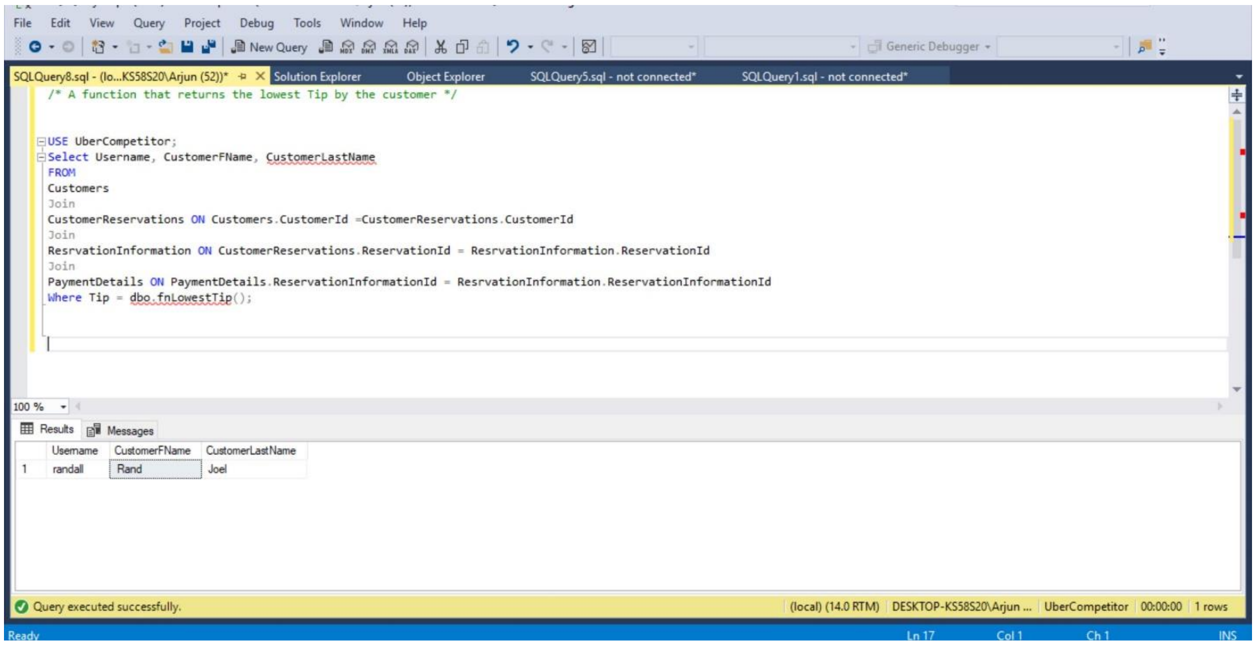
Query executed successfully. (local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 0 rows

The lowest tip given by the customer is identified using the function.

Source Code:

```
/* A function that returns the lowest Tip by the customer */

USE UberCompetitor;
Select Username, CustomerFName, CustomerLastName
FROM
Customers
Join
CustomerReservations ON Customers.CustomerId =CustomerReservations.CustomerId
Join
ResrvationInformation ON CustomerReservations.ReservationId =
ResrvationInformation.ReservationId
Join
PaymentDetails ON PaymentDetails.ReservationInformationId =
ResrvationInformation.ReservationInformationId
Where Tip = dbo.fnLowestTip();
```

Screen Shot:


File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
SQLQuery8.sql - (lo...KS58S20\Arjun (52))*
Solution Explorer
Object Explorer
SQLQuery5.sql - not connected*
SQLQuery1.sql - not connected*
100 %
Results
Messages
Ready
Ln 17
Col 1
Ch 1
INS

```
/* A function that returns the lowest Tip by the customer */

USE UberCompetitor;
Select Username, CustomerFName, CustomerLastName
FROM
Customers
Join
CustomerReservations ON Customers.CustomerId =CustomerReservations.CustomerId
Join
ResrvationInformation ON CustomerReservations.ReservationId = ResrvationInformation.ReservationId
Join
PaymentDetails ON PaymentDetails.ReservationInformationId = ResrvationInformation.ReservationInformationId
Where Tip = dbo.fnLowestTip();
```

	Username	CustomerFName	CustomerLastName
1	randall	Rand	Joel

Query executed successfully. (local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 1 rows

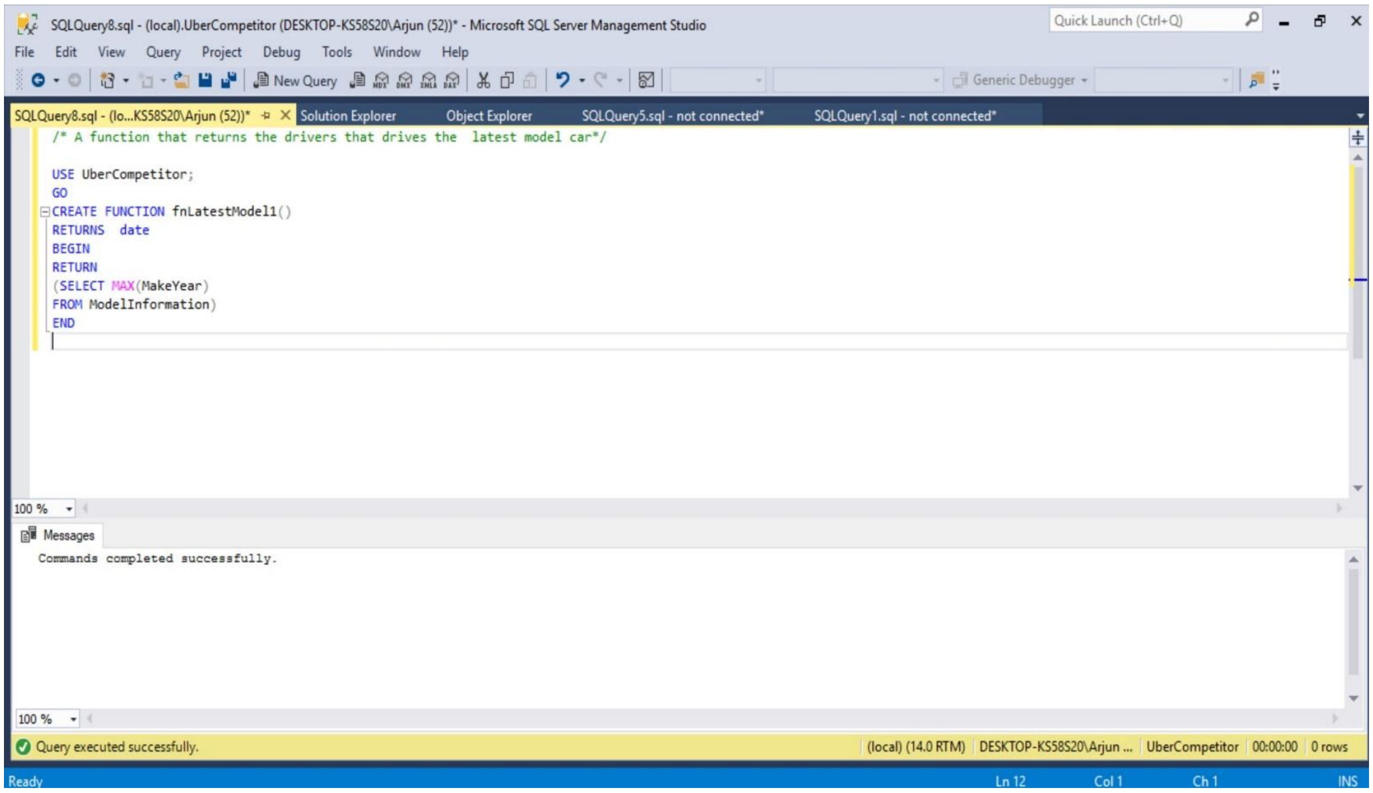
4.32

The function returns the driver with latest model car

Source Code:

```
/* A function that returns the drivers the latest model car*/

USE UberCompetitor;
GO
CREATE FUNCTION fnLatestModel1()
RETURNS  date
BEGIN
RETURN
(SELECT MAX(MakeYear)
FROM ModelInformation)
END
```

Screen Shot:



```
/* A function that returns the drivers that drives the  latest model car*/

USE UberCompetitor;
GO
CREATE FUNCTION fnLatestModel1()
RETURNS  date
BEGIN
RETURN
(SELECT MAX(MakeYear)
FROM ModelInformation)
END
```

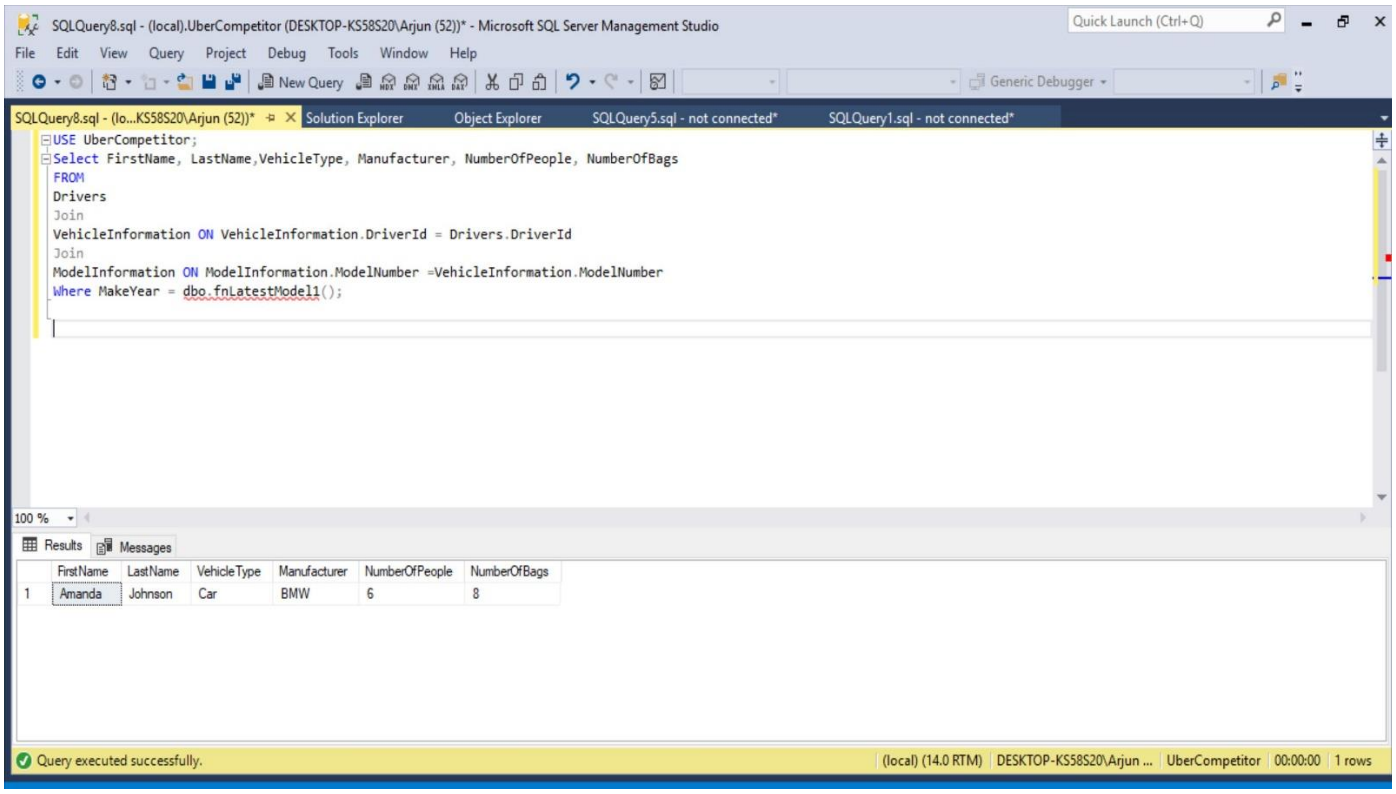
```
Commands completed successfully.
```

Query executed successfully.

Source Code:

```
USE UberCompetitor;
Select FirstName, LastName,VehicleType, Manufacturer, NumberOfPeople, NumberOfBags
FROM
Drivers
Join
VehicleInformation ON VehicleInformation.DriverId = Drivers.DriverId
Join
ModelInformation ON ModelInformation.ModelNumber =VehicleInformation.ModelNumber
Where MakeYear = dbo.fnLatestModel1();
```

Screen Shot:


SQLQuery8.sql - (local).UberCompetitor (DESKTOP-KS58S20\Arjun (52))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
SQLQuery8.sql - (lo...KS58S20\Arjun (52))*
Solution Explorer
Object Explorer
SQLQuery5.sql - not connected*
SQLQuery1.sql - not connected*
100 %
Results
Messages

```
USE UberCompetitor;
Select FirstName, LastName,VehicleType, Manufacturer, NumberOfPeople, NumberOfBags
FROM
Drivers
Join
VehicleInformation ON VehicleInformation.DriverId = Drivers.DriverId
Join
ModelInformation ON ModelInformation.ModelNumber =VehicleInformation.ModelNumber
Where MakeYear = dbo.fnLatestModel1();
```

	FirstName	LastName	VehicleType	Manufacturer	NumberOfPeople	NumberOfBags
1	Amanda	Johnson	Car	BMW	6	8

Query executed successfully. (local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 1 rows

4.4. Triggers

Triggers are created and tested in the database.

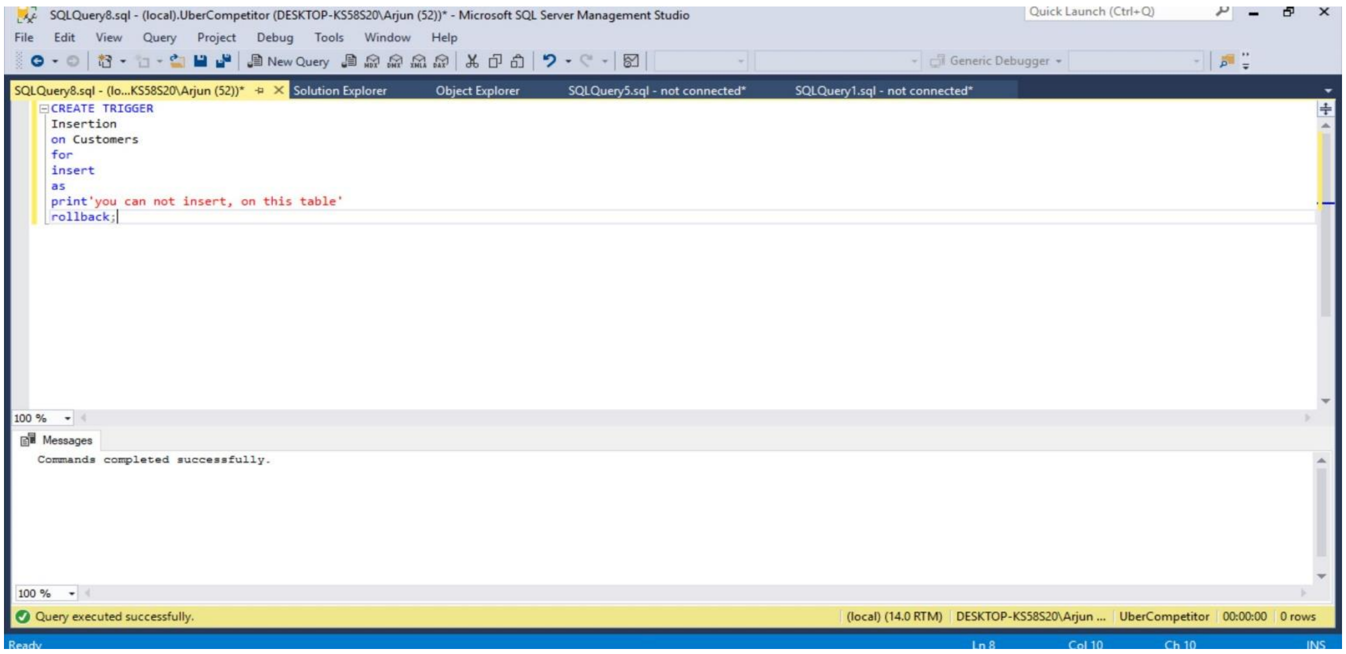
4.41

A trigger that does not allow to insert more rows on the table.

Source code:

```
CREATE TRIGGER
Insertion
on Customers
for
insert
as
print'you can not insert, on this table'
rollback;
```

Screen Shot:



SQLQuery8.sql - (local).UberCompetitor (DESKTOP-KS58S20\Arjun (52))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
SQLQuery8.sql - (lo...KS58S20\Arjun (52))*
Solution Explorer
Object Explorer
SQLQuery5.sql - not connected*
SQLQuery1.sql - not connected*
100 %
Messages
Ready
Ln 8
Col 10
Ch 10
INS

```
CREATE TRIGGER
Insertion
on Customers
for
insert
as
print'you can not insert, on this table'
rollback;
```

```
Commands completed successfully.
```

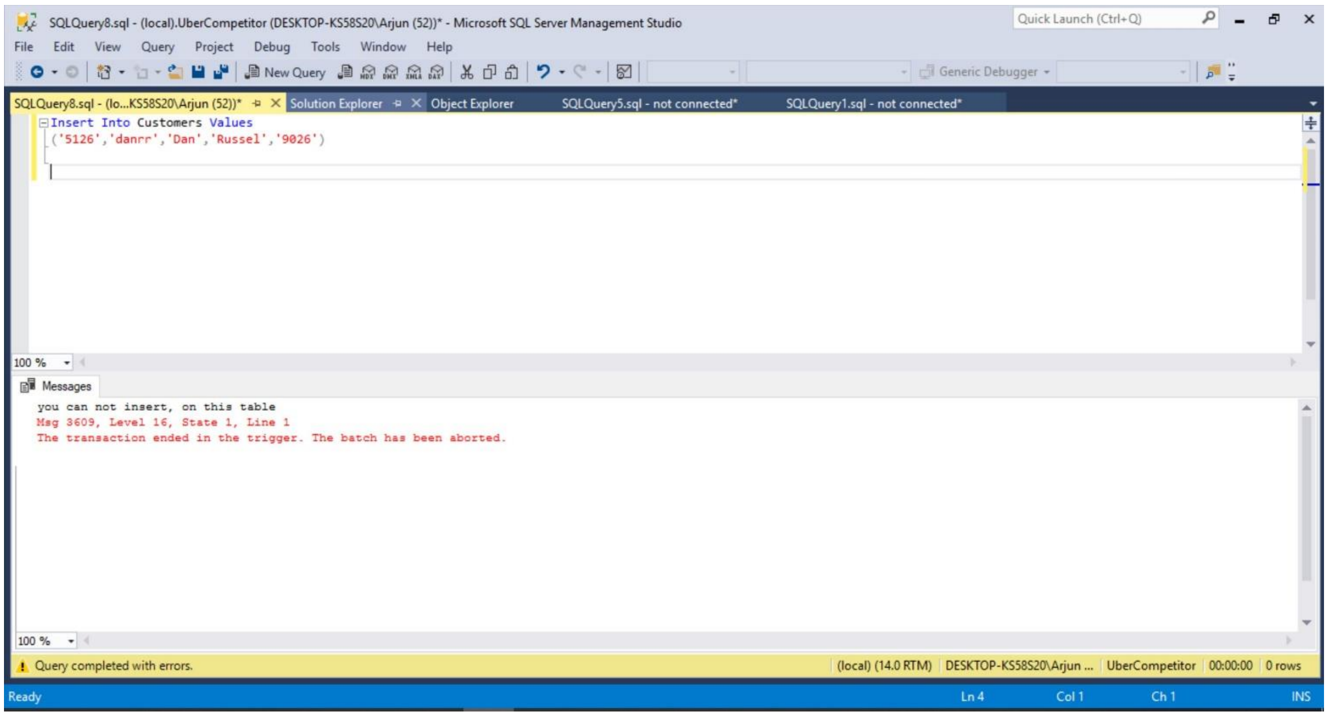
Query executed successfully. (local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 0 rows

Trigger Execution

Source Code:

```
Insert Into Customers Values
('5126','danrr','Dan','Russel','9026')
```

Screen shot:



SQLQuery8.sql - (local).UberCompetitor (DESKTOP-KS58S20\Arjun (52))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
SQLQuery8.sql - (lo...KS58S20\Arjun (52))*
Solution Explorer
Object Explorer
SQLQuery5.sql - not connected*
SQLQuery1.sql - not connected*
100 %
Messages
100 %
Query completed with errors.
(local) (14.0 RTM)
DESKTOP-KS58S20\Arjun ...
UberCompetitor
00:00:00
0 rows
Ready
Ln 4
Col 1
Ch 1
INS

```
Insert Into Customers Values
('5126','danrr','Dan','Russel','9026')
```

```
you can not insert, on this table
Msg 3609, Level 16, State 1, Line 1
The transaction ended in the trigger. The batch has been aborted.
```

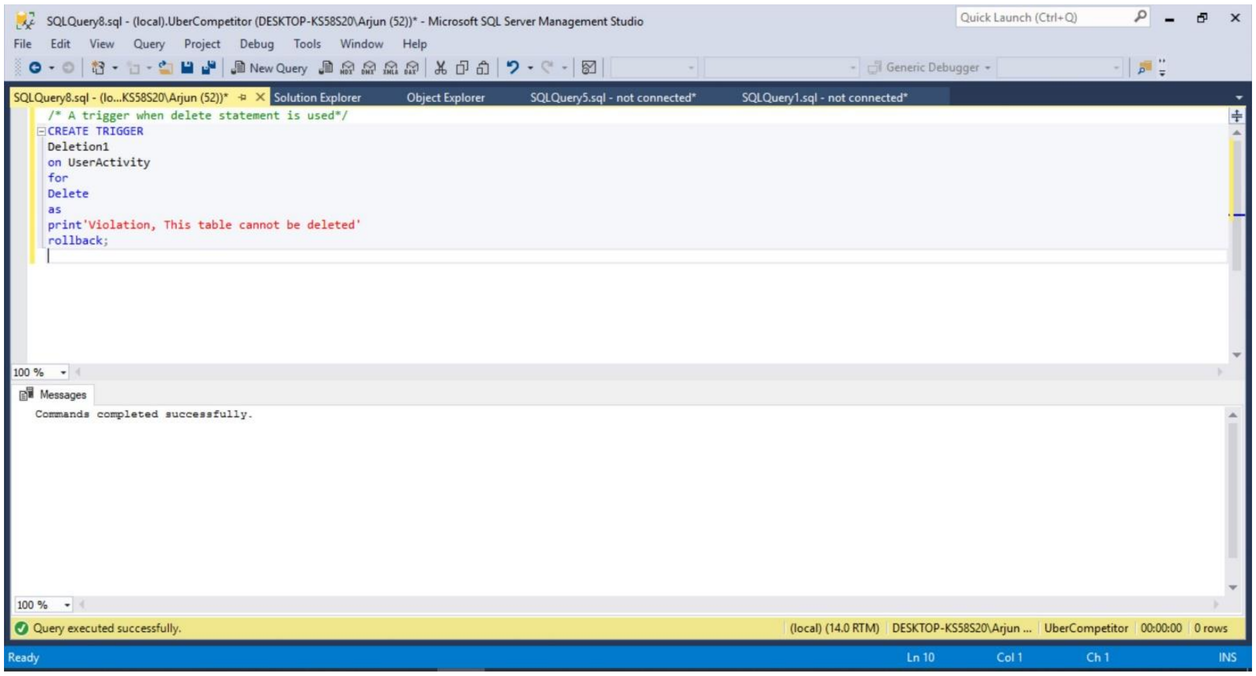
4.42

Here a trigger is created when the delete query is used

Source Code:

```
/* A trigger when delete query is used*/
CREATE TRIGGER
Deletion1
on UserActivity
for
Delete
as
print'Violation, This table cannot be deleted'
rollback;
```

Screen Shot:



```
/* A trigger when delete statement is used*/
CREATE TRIGGER
Deletion1
on UserActivity
for
Delete
as
print'Violation, This table cannot be deleted'
rollback;
```

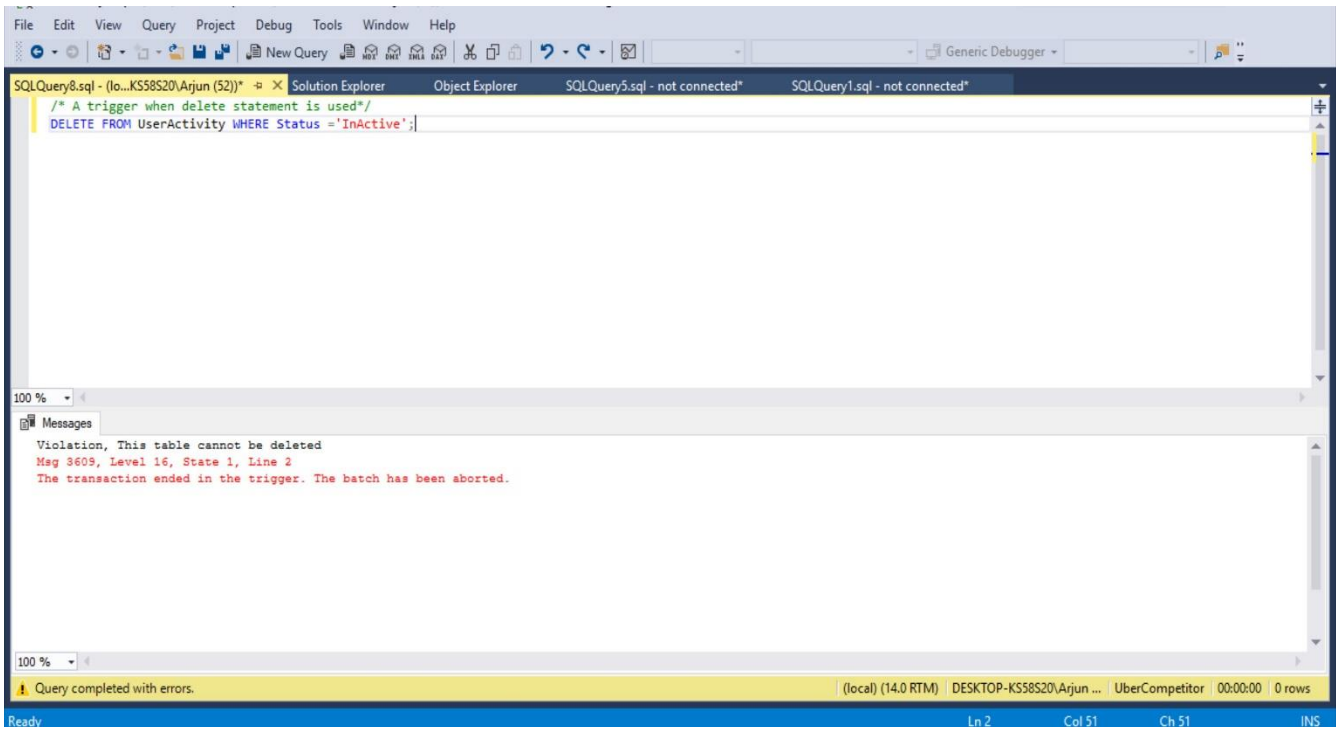
```
Commands completed successfully.
```

Query executed successfully.

Source Code:

```
/* A trigger when delete statement is used*/
DELETE FROM UserActivity WHERE Status ='InActive';
```

Screen Shot:



```
/* A trigger when delete statement is used*/
DELETE FROM UserActivity WHERE Status ='InActive';
```

```
Violation, This table cannot be deleted
Msg 3609, Level 16, State 1, Line 2
The transaction ended in the trigger. The batch has been aborted.
```

Query completed with errors. | (local) (14.0 RTM) | DESKTOP-KS58S20\Arjun ... | UberCompetitor | 00:00:00 | 0 rows

4.5 Security Levels

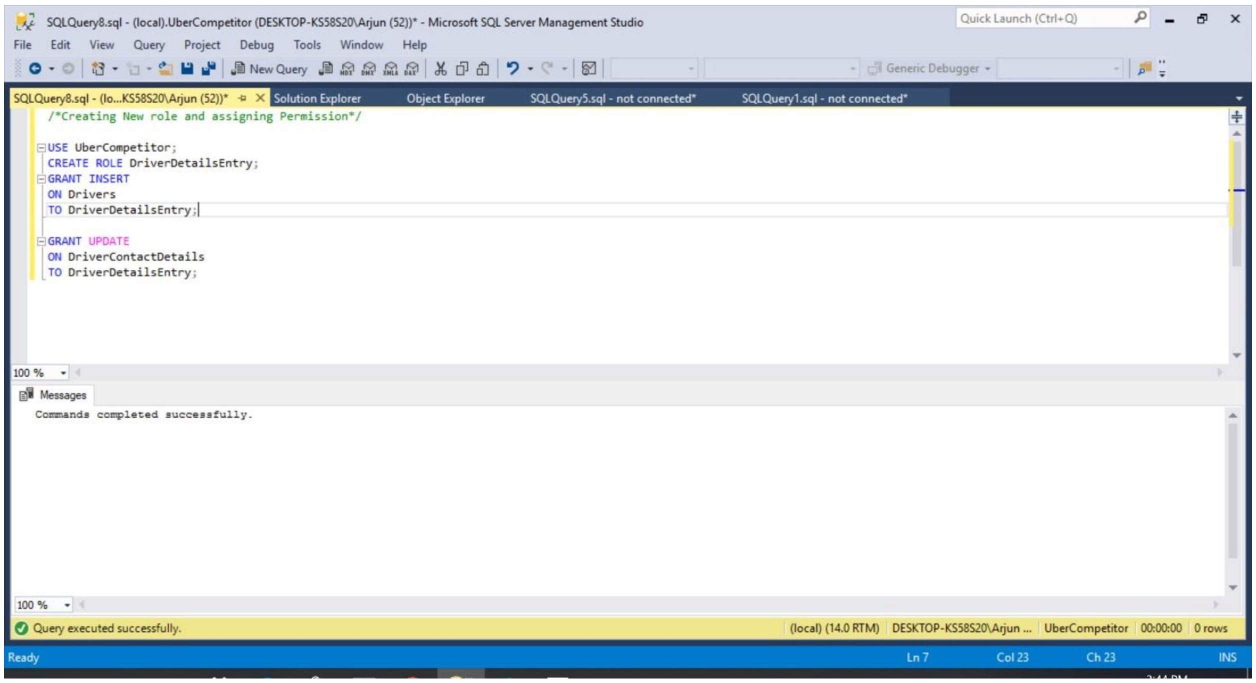
Security levels are created, and roles are added to users and also logins are created

4.5.1 Creating Roles

Source Code:

```
/*Creating New role and assigning Permission*/

USE UberCompetitor;
CREATE ROLE DriverDetailsEntry;
GRANT INSERT
ON Drivers
TO DriverDetailsEntry;

GRANT UPDATE
ON DriverContactDetails
TO DriverDetailsEntry
```

Screen Shot:



SQLQuery8.sql - (local).UberCompetitor (DESKTOP-KS58S20\Arjun (52))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
SQLQuery8.sql - (lo...KS58S20\Arjun (52))*
Solution Explorer
Object Explorer
SQLQuery5.sql - not connected*
SQLQuery1.sql - not connected*
100 %
Messages
Query executed successfully.
(local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 0 rows
Ready Ln 7 Col 23 Ch 23 INS

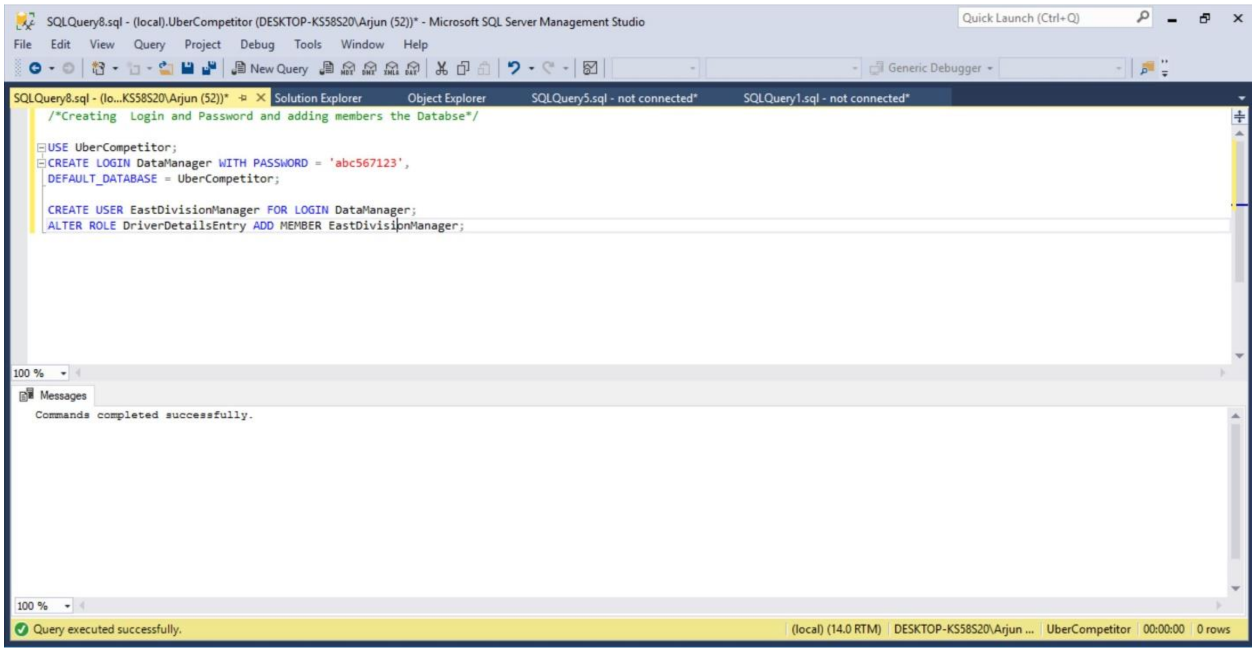
```
/*Creating New role and assigning Permission*/

USE UberCompetitor;
CREATE ROLE DriverDetailsEntry;
GRANT INSERT
ON Drivers
TO DriverDetailsEntry;

GRANT UPDATE
ON DriverContactDetails
TO DriverDetailsEntry;
```

```
Commands completed successfully.
```

Source Code:

```
/*Creating  Login and Password and adding members the Databse*/

USE UberCompetitor;
CREATE LOGIN DataManager WITH PASSWORD = 'abc567123',
DEFAULT_DATABASE = UberCompetitor;

CREATE USER EastDivisionManager FOR LOGIN DataManager;
ALTER ROLE DriverDetailsEntry ADD MEMBER EastDivisionManager;
```

Screen Shot:


SQLQuery8.sql - (local).UberCompetitor (DESKTOP-KS58S20\Arjun (52))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
SQLQuery8.sql - (lo...KS58S20\Arjun (52))*
Solution Explorer
Object Explorer
SQLQuery5.sql - not connected*
SQLQuery1.sql - not connected*
100 %
Messages

```
/*Creating  Login and Password and adding members the Databse*/

USE UberCompetitor;
CREATE LOGIN DataManager WITH PASSWORD = 'abc567123',
DEFAULT_DATABASE = UberCompetitor;

CREATE USER EastDivisionManager FOR LOGIN DataManager;
ALTER ROLE DriverDetailsEntry ADD MEMBER EastDivisionManager;
```

```
Commands completed successfully.
```

Query executed successfully. (local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 0 rows

4.6

Creating Views

A number of views with various conditions are created and tested in this database

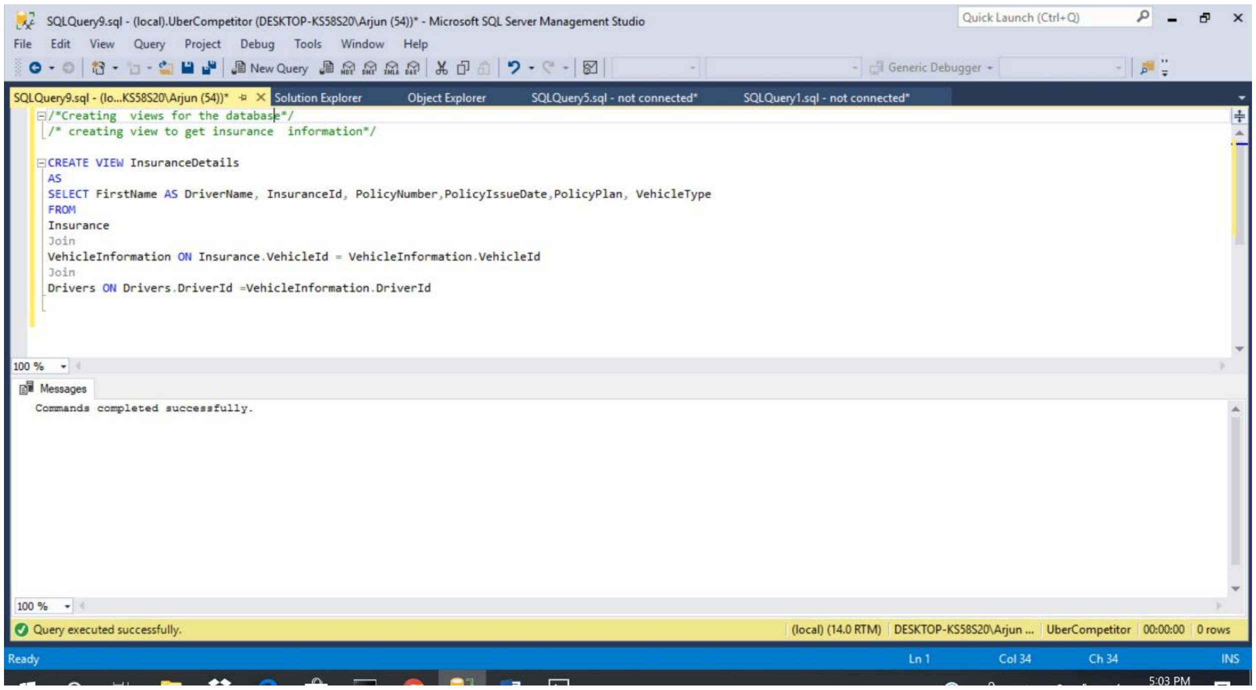
4.6.1

A view is created to get the insurance information

Source Code:

```
/*Creating  views for the datbase*/
/* creating view to get insurance  information*/

CREATE VIEW InsuranceDetails
AS
SELECT FirstName AS DriverName, InsuranceId, PolicyNumber,PolicyIssueDate,PolicyPlan,
VehicleType
FROM
Insurance
Join
VehicleInformation ON Insurance.VehicleId = VehicleInformation.VehicleId
Join
Drivers ON Drivers.DriverId =VehicleInformation.DriverId
```

Screen Shot:



```
/*Creating  views for the database*/
/* creating view to get insurance  information*/

CREATE VIEW InsuranceDetails
AS
SELECT FirstName AS DriverName, InsuranceId, PolicyNumber,PolicyIssueDate,PolicyPlan, VehicleType
FROM
Insurance
Join
VehicleInformation ON Insurance.VehicleId = VehicleInformation.VehicleId
Join
Drivers ON Drivers.DriverId =VehicleInformation.DriverId
```

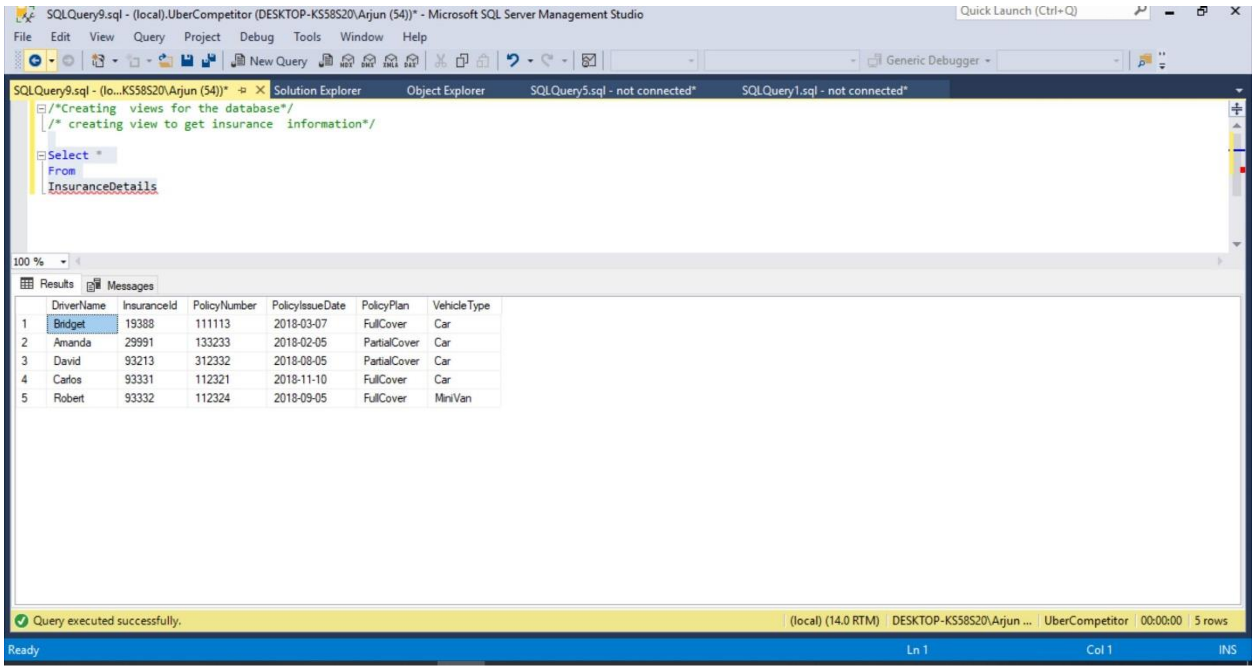
```
Commands completed successfully.
```

Query executed successfully.

Source Code:

```
Select *
From
InsuranceDetails
```

Screen Shot:



```
/*Creating  views for the database*/
/* creating view to get insurance  information*/

Select *
From
InsuranceDetails
```

	DriverName	InsuranceId	PolicyNumber	PolicyIssueDate	PolicyPlan	VehicleType
1	Bridget	19388	111113	2018-03-07	FullCover	Car
2	Amanda	29991	133233	2018-02-05	PartialCover	Car
3	David	93213	312332	2018-08-05	PartialCover	Car
4	Carlos	93331	112321	2018-11-10	FullCover	Car
5	Robert	93332	112324	2018-09-05	FullCover	MiniVan

Query executed successfully. (local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 5 rows

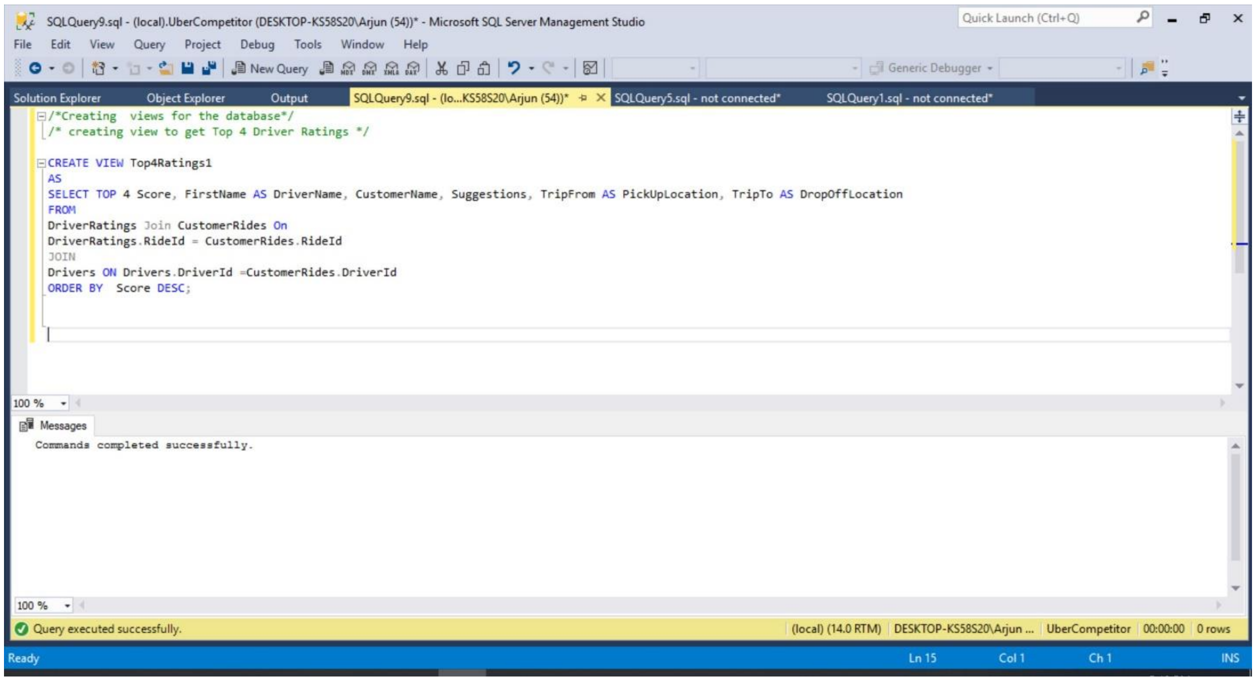
4.6.2

Creating a view to get the top 4 driver ratings

Souce Code:

```
Creating  views for the database*/
/* creating view to get Top 4 Driver Ratings */

CREATE VIEW Top4Ratings1
AS
SELECT TOP 4 Score, FirstName AS DriverName, CustomerName, Suggestions, TripFrom AS
PickUpLocation, TripTo AS DropOffLocation
FROM
DriverRatings Join CustomerRides On
DriverRatings.RideId = CustomerRides.RideId
JOIN
Drivers ON Drivers.DriverId =CustomerRides.DriverId
ORDER BY SCORE DESC;
```

Screen Shot:


SQLQuery9.sql - (local).UberCompetitor (DESKTOP-KS58S20\Arjun (54))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
Solution Explorer Object Explorer Output
SQLQuery9.sql - (lo...KS58S20\Arjun (54))*
SQLQuery5.sql - not connected*
SQLQuery1.sql - not connected*
100 %
Messages

```
/*Creating  views for the database*/
/* creating view to get Top 4 Driver Ratings */

CREATE VIEW Top4Ratings1
AS
SELECT TOP 4 Score, FirstName AS DriverName, CustomerName, Suggestions, TripFrom AS PickUpLocation, TripTo AS DropOffLocation
FROM
DriverRatings Join CustomerRides On
DriverRatings.RideId = CustomerRides.RideId
JOIN
Drivers ON Drivers.DriverId =CustomerRides.DriverId
ORDER BY  Score DESC;
```

```
Commands completed successfully.
```

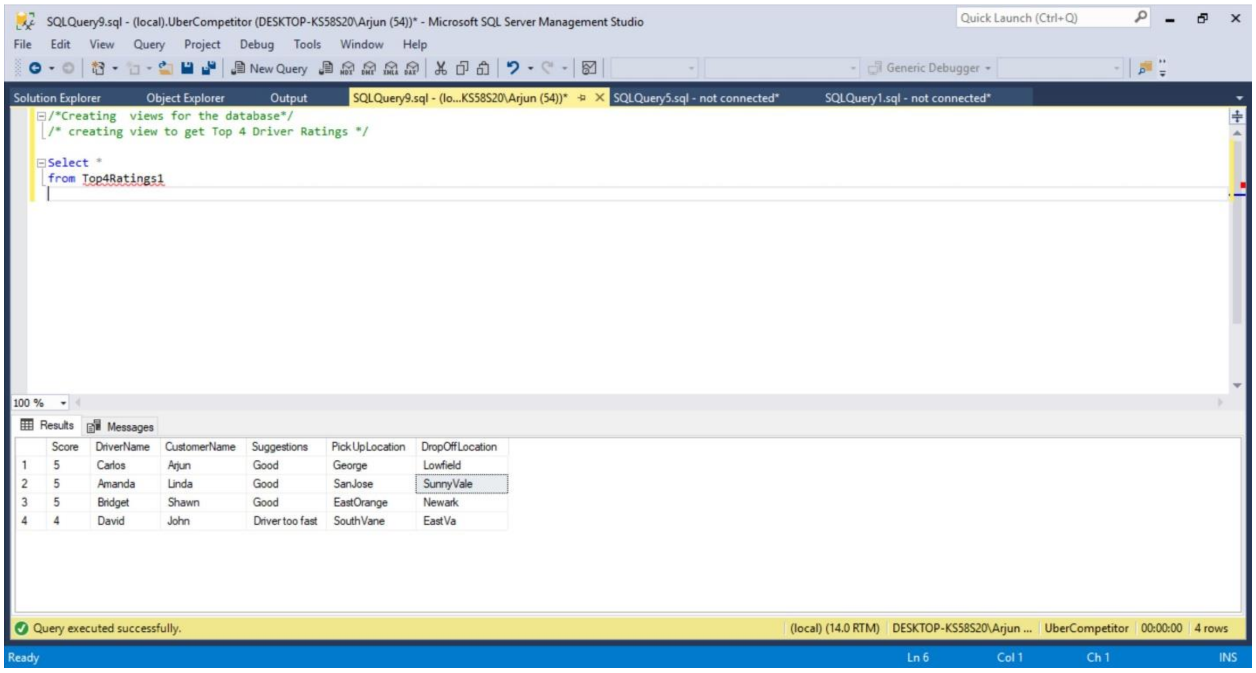
Query executed successfully. (local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 0 rows

Ready Ln 15 Col 1 Ch 1 INS

Source Code:

```
Select *
From
Top4Ratings1
```

Screen Shot:



SQLQuery9.sql - (local).UberCompetitor (DESKTOP-KS58S20\Arjun (54))* - Microsoft SQL Server Management Studio

```
/*Creating  views for the database*/
/* creating view to get Top 4 Driver Ratings */

Select *
from Top4Ratings1
```

	Score	DriverName	CustomerName	Suggestions	PickUpLocation	DropOffLocation
1	5	Carlos	Arjun	Good	George	Lowfield
2	5	Amanda	Linda	Good	SanJose	SunnyVale
3	5	Bridget	Shawn	Good	EastOrange	Newark
4	4	David	John	Driver too fast	SouthVane	EastVa

Query executed successfully. (local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 4 rows

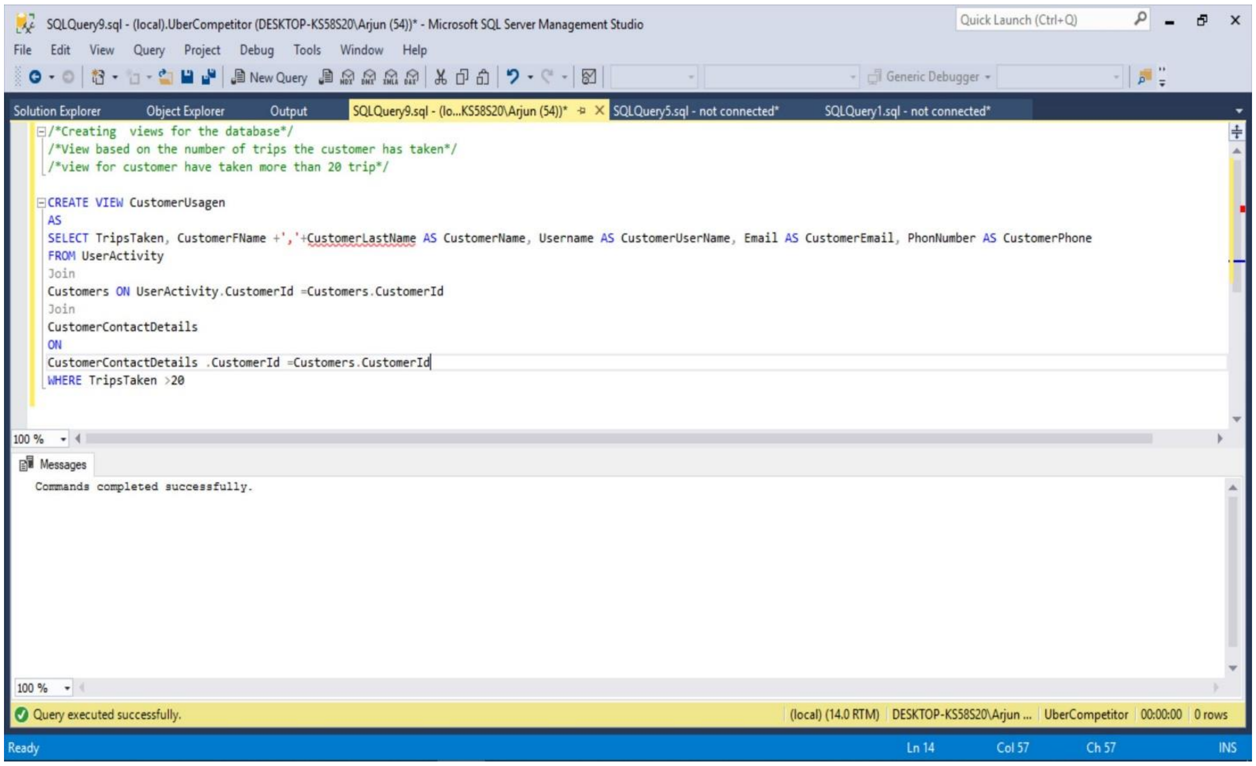
4.6.3

Creating a view to find the customers who have taken a trip more than 20 times

Source Code:

```sql
/*Creating  views for the database*/
/*View based on the number of trips the customer has taken*/
/*view for customer have taken more than 20 trip*/

CREATE VIEW CustomerUsagen
AS
SELECT TripsTaken, CustomerFName +','+CustomerLastName AS CustomerName, Username AS
CustomerUserName, Email AS CustomerEmail, PhonNumber AS CustomerPhone
FROM UserActivity
Join
Customers ON UserActivity.CustomerId =Customers.CustomerId
Join
CustomerContactDetails
ON
CustomerContactDetails .CustomerId =Customers.CustomerId
WHERE TripsTaken >20
```

Screen Shot:


SQLQuery9.sql - (local).UberCompetitor (DESKTOP-KS58S20\Arjun (54))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
Solution Explorer
Object Explorer
Output
SQLQuery9.sql - (lo...KS58S20\Arjun (54))*
SQLQuery5.sql - not connected*
SQLQuery1.sql - not connected*
100 %
Messages

```sql
/*Creating  views for the database*/
/*View based on the number of trips the customer has taken*/
/*view for customer have taken more than 20 trip*/

CREATE VIEW CustomerUsagen
AS
SELECT TripsTaken, CustomerFName +','+CustomerLastName AS CustomerName, Username AS CustomerUserName, Email AS CustomerEmail, PhonNumber AS CustomerPhone
FROM UserActivity
Join
Customers ON UserActivity.CustomerId =Customers.CustomerId
Join
CustomerContactDetails
ON
CustomerContactDetails .CustomerId =Customers.CustomerId
WHERE TripsTaken >20
```

```
Commands completed successfully.
```

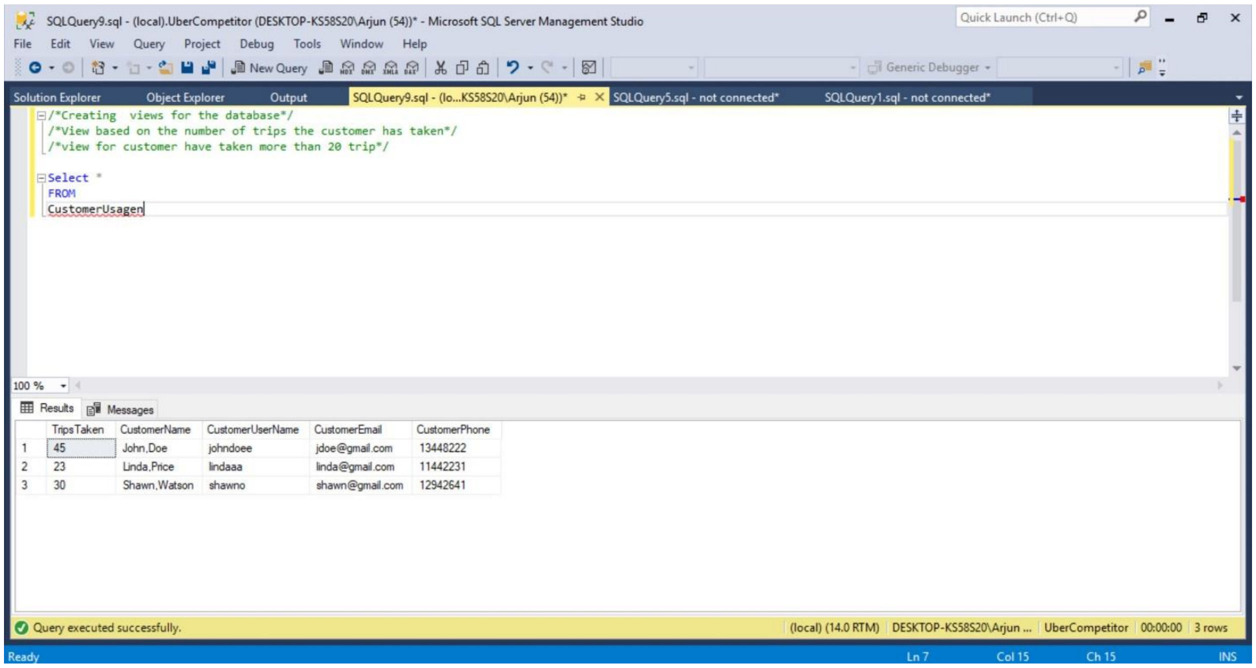
Query executed successfully. | (local) (14.0 RTM) | DESKTOP-KS58S20\Arjun ... | UberCompetitor | 00:00:00 | 0 rows

Ready | Ln 14 | Col 57 | Ch 57 | INS

Source Code:

```
Select *
FROM
CustomerUsagen
```

Screen Shot:



```
/*Creating  views for the database*/
/*View based on the number of trips the customer has taken*/
/*view for customer have taken more than 20 trip*/

Select *
FROM
CustomerUsagen
```

	TripsTaken	CustomerName	CustomerUserName	CustomerEmail	CustomerPhone
1	45	John,Doe	johndoe	jdoe@gmail.com	13448222
2	23	Linda,Price	lindaaa	linda@gmail.com	11442231
3	30	Shawn,Watson	shawno	shawn@gmail.com	12942641

Query executed successfully. (local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 3 rows

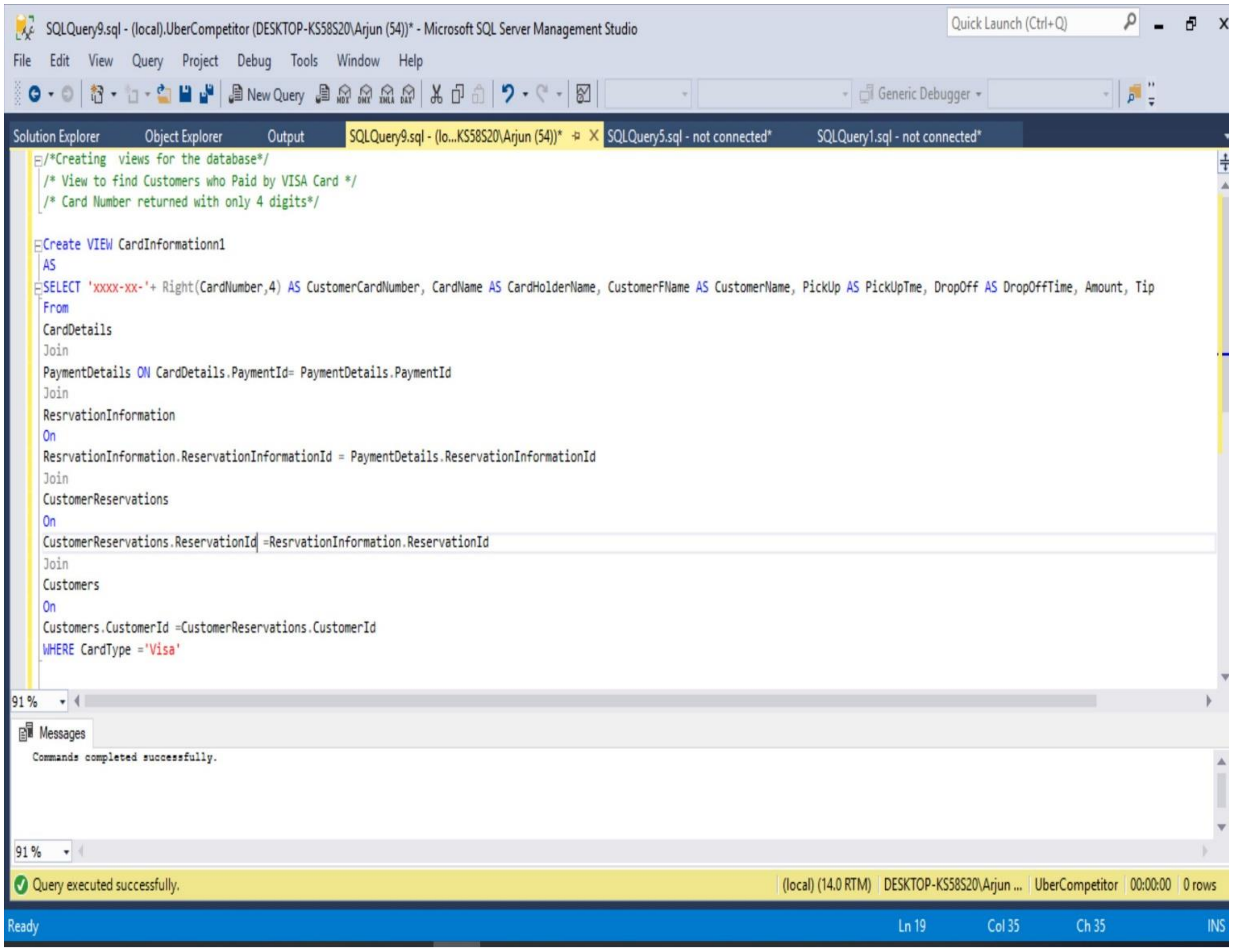
4.6.4

Creating a view to find customers who paid by visa card and the card number is returned only with 4 digits.

Source Code:

```
/*Creating  views for the database*/
/* View to find Customers who Paid by VISA Card */
/* Card Number returned with only 4 digits*/

Create VIEW CardInformationn1
AS
SELECT 'xxxx-xx-'+ Right(CardNumber,4) AS CustomerCardNumber, CardName AS CardHolderName,
CustomerFName AS CustomerName, PickUp AS PickUpTme, DropOff AS DropOffTime, Amount, Tip
From
CardDetails
Join
PaymentDetails ON CardDetails.PaymentId= PaymentDetails.PaymentId
Join
ResrvationInformation
On
ResrvationInformation.ReservationInformationId = PaymentDetails.ReservationInformationId
Join
CustomerReservations
On
CustomerReservations.ReservationId =ResrvationInformation.ReservationId
Join
Customers
On
Customers.CustomerId =CustomerReservations.CustomerId
WHERE CardType ='Visa'
```

Screen Shot:



SQLQuery9.sql - (local).UberCompetitor (DESKTOP-KS58S20\Arjun (54))* - Microsoft SQL Server Management Studio
Quick Launch (Ctrl+Q)
File Edit View Query Project Debug Tools Window Help
New Query
Generic Debugger
Solution Explorer Object Explorer Output SQLQuery9.sql - (lo...KS58S20\Arjun (54))* SQLQuery5.sql - not connected* SQLQuery1.sql - not connected*

```
/*Creating  views for the database*/
/* View to find Customers who Paid by VISA Card */
/* Card Number returned with only 4 digits*/

Create VIEW CardInformationn1
AS
SELECT 'xxxx-xx-'+ Right(CardNumber,4) AS CustomerCardNumber, CardName AS CardHolderName, CustomerFName AS CustomerName, PickUp AS PickUpTme, DropOff AS DropOffTime, Amount, Tip
From
CardDetails
Join
PaymentDetails ON CardDetails.PaymentId= PaymentDetails.PaymentId
Join
ResrvationInformation
On
ResrvationInformation.ReservationInformationId = PaymentDetails.ReservationInformationId
Join
CustomerReservations
On
CustomerReservations.ReservationId =ResrvationInformation.ReservationId
Join
Customers
On
Customers.CustomerId =CustomerReservations.CustomerId
WHERE CardType ='Visa'
```

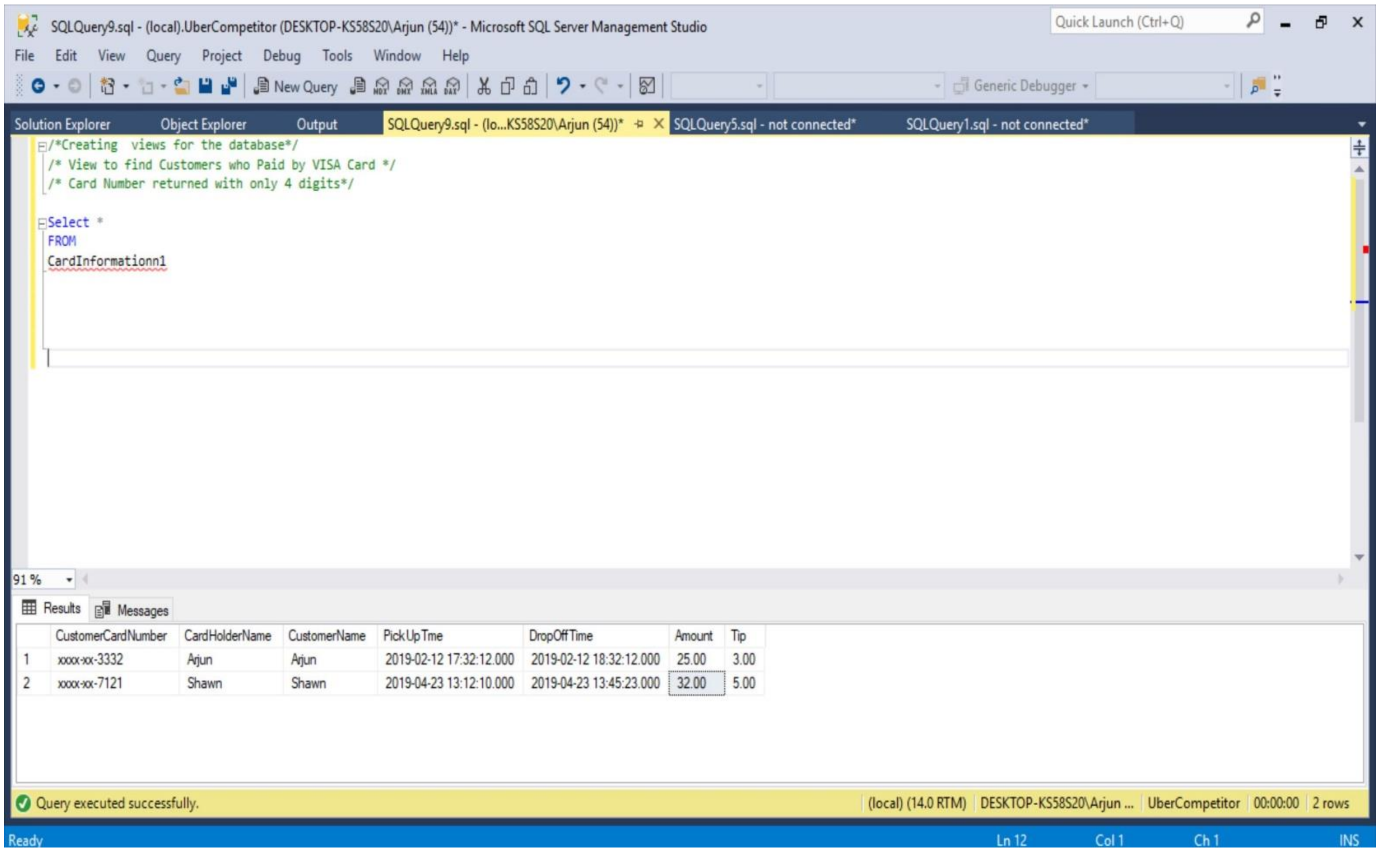
```
Commands completed successfully.
```

91 %
Query executed successfully.
(local) (14.0 RTM) | DESKTOP-KS58S20\Arjun ... | UberCompetitor | 00:00:00 | 0 rows
Ready Ln 19 Col 35 Ch 35 INS

Source Code:

```
Select *
FROM
CardInformationn1
```

Screen Shot:



```
/*Creating  views for the database*/
/* View to find Customers who Paid by VISA Card */
/* Card Number returned with only 4 digits*/

Select *
FROM
CardInformationn1
```

	CustomerCardNumber	CardHolderName	CustomerName	PickUpTme	DropOffTime	Amount	Tip
1	xxxx-xx-3332	Arjun	Arjun	2019-02-12 17:32:12.000	2019-02-12 18:32:12.000	25.00	3.00
2	xxxx-xx-7121	Shawn	Shawn	2019-04-23 13:12:10.000	2019-04-23 13:45:23.000	32.00	5.00

Query executed successfully. (local) (14.0 RTM) DESKTOP-KS58S20\Arjun ... UberCompetitor 00:00:00 2 rows

5. Conclusion

5.1 Project analysis:

The Project Uber competitor database has been created and various tables have been created. Sample data was input to the database and it was tested using stored procedures, function, views, triggers and various constraints have been used. Overall the working of the project gives the required outcome. The database was normalized and there is no redundancy in it this makes the database simple and efficient to use. The project has been designed in such a way that it could meet future requirements.

5.2 Remarks:

The Uber competitor database introduces all the concepts of creating, implementing and testing a database. The labs and the concepts introduced in class helped in developing the database.